Classification of bone defects using natural and synthetic X-ray images

Akash Roy Choudhury

School of Science

Thesis submitted for examination for the degree of Master of Science in Technology.

Espoo 23.04.2021

Thesis supervisor:

Prof. Simo Särkkä

Thesis advisors:

Dr. Leo Kärkkäinen

M.Sc. Joel Jaskari

Aalto University
School of Science

AALTO UNIVERSITY
SCHOOL OF SCIENCE

ABSTRACT OF THE
MASTER'S THESIS

Author: Akash Roy Choudhury
Title: Classification of bone defects using natural and synthetic X-ray images
Date: 23.04.2021 Language: English Number of pages: 6+69

Department of Computer Science
Code of Major: SCI3042

Supervisor: Prof. Simo Särkkä
Advisors: Dr. Leo Kärkkäinen, M.Sc. Joel Jaskari

In this thesis, we study methods to reduce the amount of data needed to create deep learning models that can detect defects in bones from X-ray images. Detecting defects in bones from X-ray images and properly annotating the images is the paramount step when it comes to corrective surgeries of bones. Annotations or labels, such as radial inclination and volar tilt are measurements that are necessary for many corrective surgeries. Generating these annotations is an arduous and manual task for medical professionals. By being able to automate the process of generating these annotations, it will be possible to reduce a significant amount of labor of these professionals.

Modern deep learning models are heavily reliant upon availability of a large amount of properly labeled data for their training. In this thesis, we experimented to find methods to create appropriate synthetic data that can be combined with natural data to train deep learning models. We designed three deep learning models to generate two different forms of annotations. The first goal was to use cycle consistent generative adversarial networks to create proper synthetic images. Then we used the synthetic images to improve classifier models that can detect defects in bones. In the end, we expanded the cycle consistent generative adversarial network so that it can accommodate three input domains instead of two and called it multi-cycleGAN. We used multi-cycleGAN to segment bones from natural X-ray images.

Our experiments concluded that by adding proper synthetic images with natural images, we can improve the performance of classifiers significantly and circumvent the persistent issue of unavailability of data. However, the multi-cycleGAN model did not generate a very accurate segmentation of bones. It was able to segment bones of forearm better than bones of wrist. It was able to understand the overall shape and positioning of the wrists in X-ray images but it did not produce proper segmentations of the individual fingers.

Keywords: deep learning, generative adversarial network, classification, X-ray, supervised learning, unsupervised learning, image processing, medical image analysis

Acknowledgements

I want to thank Professor Simo Särkkä for taking the invidious role of my supervisor. I also want to thank Professor Leo Kärkkäinen and Joel Jaskari for agreeing to be my thesis advisors and for their continuous support throughout the entire project. Without their guidance it would have been impossible for me to complete the project. I am highly obliged to them for keeping me motivated and for always pointing out my mistakes and guiding me towards the right direction.

Otaniemi, 23.04.2021

Akash Roy Choudhury

Contents

Abbreviations

Adam	adaptive moment estimation
CGAN	conditional generative adversarial networks
CLAHE	contrast limited adaptive histogram equalization
CNN	convolutional neural networks
CycleGAN	cycle-consistent generative adversarial networks
DCGAN	deep convolutional generative adversarial networks
DNN	deep neural networks
EM	electromagnetic
GAN	generative adversarial networks
GPU	graphical processing unit
MLP	multi-layer perceptron
MNIST	modified national institute of standards and technology
Multi-CycleGAN	multi domain cycle-consistent generative adversarial networks
MRI	magnetic resonance imaging
MSE	mean squared error
OCT	optical coherence tomography
ReLU	rectified linear unit
SGD	stochastic gradient descent
SSD	single shot multibox detector
YOLO	you only look once

1 Introduction

The results of a study conducted by the Harvard School of Public Health in 2013 show that at least 43 million injuries are caused by the lack of medical care and misdiagnosis every year [Jha et al. (2013)]. A loss of 23 million years of healthy lives ensues from this and around two-thirds of these stem from low or middle-income countries. The study indicated that this number will grow every year and pointed out that a possible course of action is increasing investments to promote the means of medical diagnosis and accessible healthcare.

With increasing computational power and availability of data, deep neural networks (DNN) have been able to improve significantly to accomplish medical image analysis tasks such as detection of diabetic retinopathy from retinal fundus images [Gulshan et al. (2016)], detection of pneumonia from chest X-ray images [Yadav and Jadhav (2019)], cancer detection [Cireşan et al. (2013)], brain magnetic resonance imaging (MRI) analysis [Akkus et al. (2017)], or cardiac MRI analysis [Avendi et al. (2016)]. Two of the most widely used medical image analysis techniques include image classification and image segmentation [Zhou et al. (2017)].

However, deep neural networks are heavily reliant upon availability of a large amount of properly annotated data [Donahue et al. (2014)]. Unavailability of such data leads to training of models that produce unreliable results. In the domain of medical image analysis, generating these annotations are subject to extensive cost of imaging and image labelling. The cost of imaging can be attributed to expensive imaging devices such as MRI machines or X-ray machines whereas the cost of labelling or annotating is ascribed to the reason that these annotations are often created manually by experts and are very time consuming.

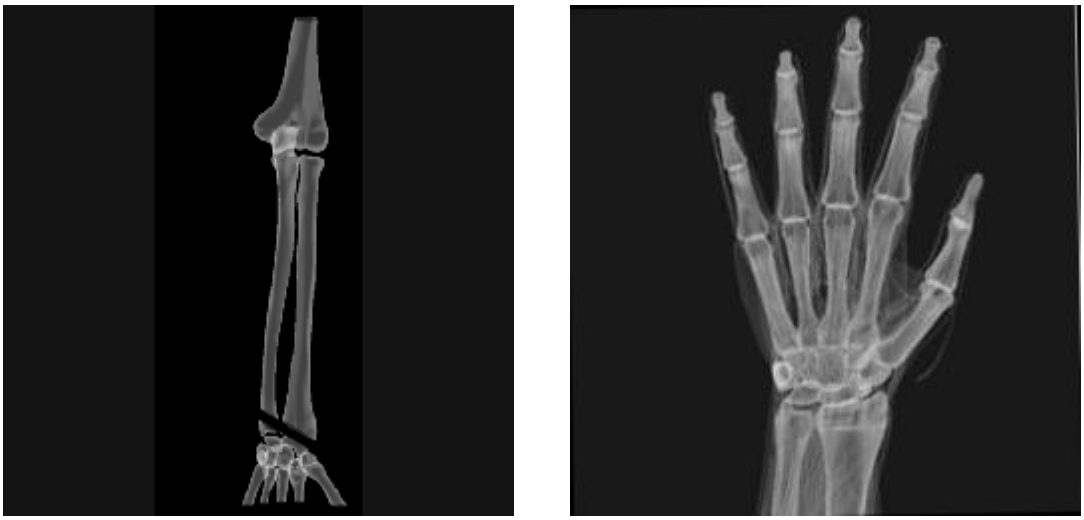
This thesis demonstrates methods to overcome the challenges of creating appropriate deep learning models due to lack of data and lack of labels or annotations for X-ray images of bones. To accomplish this task, synthetically generated X-ray images along with their proper annotations were used. The purpose of this thesis was to accomplish the following tasks:

1. Develop deep learning model to create natural looking X-ray images out of synthetic X-ray images.

2. Use the transformed synthetic images to improve deep learning models that can identify defective bones from X-ray images.

3. Use synthetic X-ray images to develop deep learning models that can generate annotations from natural X-ray images.

To accomplish the challenges described above the thesis must identify appropriate datasets that can be used for experimentation, then use the data to train appropriate deep learning models to label X-ray images with bone defects and fine tune the models and evaluate the models. Finally, the thesis must contribute towards finding a method to generate annotations from natural X-ray images which are not accompanied by corresponding annotations. Examples of synthetically generated X-ray images are



(a) Abnormal synthetic forearm (b) Normal synthetic wrist

Figure 1: Examples of synthetic images.

shown in Figure 1. Figure 1a is an example of synthetic image of forearm with bone defects and Figure 1b is an example of synthetic image of wrist with no bone defects.

The rest of the thesis is organized as follows. Chapter 2 reviews the basic principles of digital image processing and medical image analysis. Then it goes on to provide basic mathematical foundations and central concepts of deep neural networks and generative adversarial networks which are used in the thesis. In the end, this chapter provides with literature review of synthetic medical image generation. Chapter 3 provides details about the datasets used and the experiments conducted. In Chapter 4, the results of the conducted experiments are presented. Chapter 5 concludes the thesis by discussing the results of the experiments, their implications, limitations of the current work, and proposes methods to further develop the models.

2 Background

2.1 Digital Image Processing

Retina in human eyes can capture electromagnetic (EM) radiation with wavelengths between 400 and 700 nanometer and that is what we know as visible light [Suetens (2017)]. Our brain processes the absorbed radiation and forms images. Gonzalez and Woods (2008) defined an image as a function with two variables $f(x, y)$ where x and y are spatial coordinates and the output of the function $f(.)$ is the value of intensity at that point. A digital image is a variant of image where the values of x,y and $f(x, y)$ are all finite and discrete [Stockman and Shapiro (2001)]. The research area that studies changes in images when it goes through linear or non-linear operations by means of a digital computer is called digital image processing [Gonzalez and Woods (2008)]. Each digital image is a composition of a finite number of unit elements [Gonzalez and Woods (2008); Stockman and Shapiro (2001)]. These are called pixels which contain the information of the intensity value.

The human retina contains three types of photoreceptor cone cells whose job is to transform the incident light to different colors [Suetens (2017)]. Since there are only three types of cone receptors, three numbers representing the values of intensity of the incident light are necessary and sufficient to accurately describe any discernible color. Historically, digital images are divided in three classes [Stockman and Shapiro (2001)] based on the color they display. A grey scale image also known as monochrome digital image consists of a singular intensity value per pixel. A multispectral image has a set of values at each pixels to describe its intensity. If the image is a color image, then the set has 3 elements. A binary image, as described by the name, only has values 0 or 1 in every pixel. It should also be noted that digital images are described by using many types of coordinate systems [Stockman and Shapiro (2001)]. Raster oriented coordinate system, where rows and columns start at [0,0] from top left, cartesian coordinate system where rows and columns start at [0,0] from bottom left, and cartesian coordinate system where rows and columns start at [0,0] at the image center are examples of a few.

Figure 2 illustrates a flow diagram of the processes involved in image analysis [Umbaugh (2010)]. In preprocessing phase, unnecessary information such as noise is removed from the images that may have been added to the image during its acquisition process. In the next stage, the image is transformed from one domain to a different domain. For example, a multispectral image can be converted to a grey scale image. After this, filtering techniques can be applied to the images that further reduce the data in the image to provide us with extractable features. These features are then used for whatever analysis deemed necessary. The whole process then can be repeated in a loop until satisfactory results are obtained. If the example of automated analysis of text is considered, starting from the process of acquiring an image of the area containing the text, preprocessing that image, extracting the individual characters through filtering, describing them in a form suitable for a computer to understand by feature extraction and in the end recognising them, are the steps involved in the process [Gonzalez and Woods (2008)]. And the entire process is an

example of image analysis.

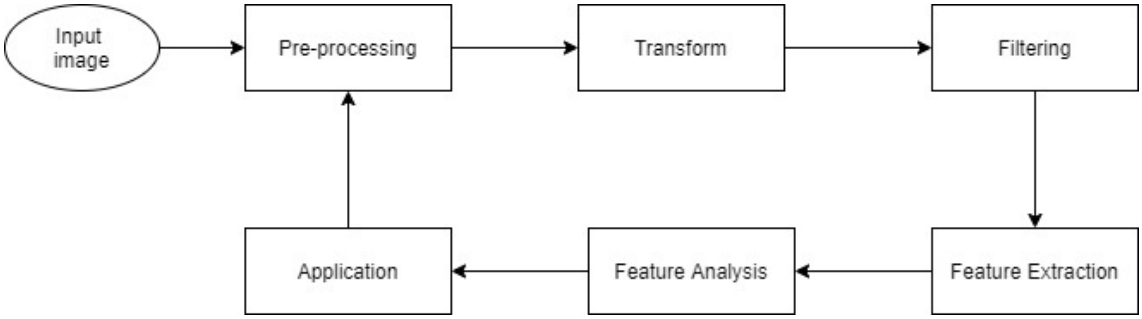


Figure 2: Processes of image analysis [Umbaugh (2010)].

2.2 Medical Image Analysis

In recent times, three-dimensional (3-D) imaging has equipped scientists to gather high quality images [Dougherty (2011)]. As a result of which, significant advances have been made in medical image analysis. Computer-aided diagnosis has emerged as one of the most important research areas in scientific imaging and has become an essential tool in early detection and diagnosis of severe medical conditions such as cancer [Dougherty (2011)]. On both microscopic (cellular level) and macroscopic (organ level) levels, multitude of medical imaging devices are used to examine human body. The general goal of medical image analysis is to extract meaningful information from the images produced from these medical devices and put those to practical use [Meyer-Bäse et al. (2004)]. However, the extraction of meaningful information requires sophisticated image processing techniques to enhance the interpretability of the information which is used to generate automatic or semi-automatic detection of ailments [Meyer-Bäse et al. (2004)]. From generating images to diagnosing diseases, Rangayyan (2004) has divided the entire process into several steps which form an iterative loop. Figure 3 illustrates the structure and order of these steps.

Imaging devices such as X-ray machines, γ-ray generators, ultrasound echo generators, nuclear magnetic resonance induction generators (also known as Magnetic resonance imaging or MRI) pass signal to patient body and gather resulting signals in analog form. These imaging devices are the most important and often most expensive equipments in the entire process [Rangayyan (2004)]. Table 1 summarises a list of these devices and the corresponding organs where they are used [Meyer-Bäse et al. (2004)]. The analog signals are then converted to digital images by digital transducers. Transducers are devices that convert one form of energy to another and digital transducers are used to convert continuous analog signals into discrete digital signal. These digital images are then stored in proper storing devices. Image processing techniques are used on the stored images for image enhancement and filtering the data. The enhanced images are then used to detect regions or objects of interest from where appropriate features are extracted. These features are then used for classification, pattern recognition, and segmentation of the images which in

turn provide the diagnostic decisions. In the end, upon reviewing the decisions, a physician can provide the remedy to the patient.

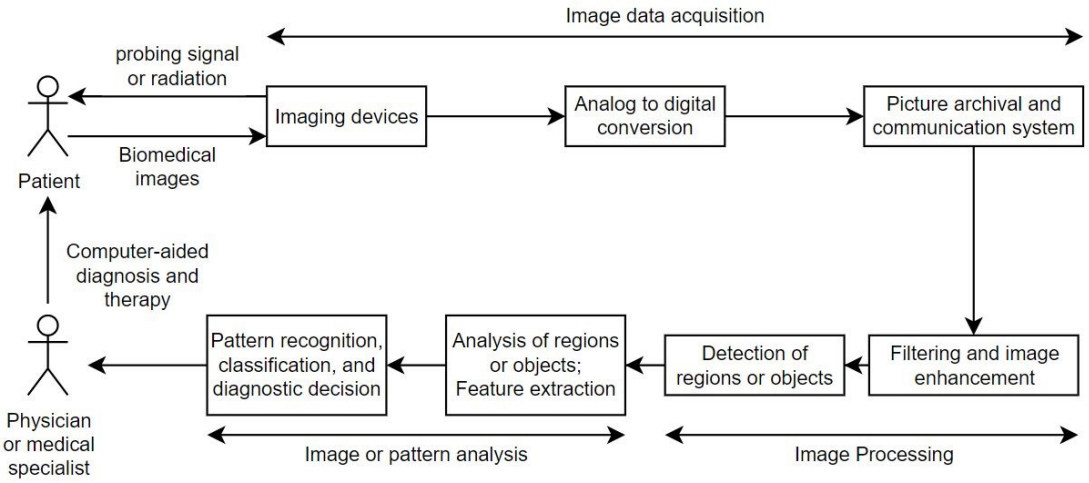


Figure 3: Steps involved in computer-aided diagnosis of diseases [Rangayyan (2004)].

Imaging device	Organs of application
X-ray generators	Breast, lung, bone
γ-ray generators	Brain, organ parenchyma, heart function
MRI	Soft tissue, disks, brain
ultrasound echo generators	Fetus, pathological changes, internal organs

Table 1: List of most important radiologic imaging devices and some organs of their application [Meyer-Bäse et al. (2004)].

In this thesis, the scope is limited to analysis of bones using X-ray images and specifically, to segmentation of bones from X-ray images and classification of normal and abnormal bones from X-ray images.

2.3 Working Principle of X-ray Machines

Since the discovery of X-rays by Wilhelm Konard Röntgen back in 1895, they have been widely used in medical diagnosis of different parts of body which include heart, lungs, blood vessels, and bones [Suetens (2017)]. Being electromagnetic in nature, X-ray radiation consists of photons. The energy of a photon can be described by the equation:

$$E = hf = \frac{hc}{\lambda}, \tag{1}$$

where E is the energy, h is the Planck's constant, c is the speed of light and λ, f are the wavelength and frequency of the photon, respectively [Suetens (2017)]. X-rays are created in a vacuum tube which consists of a cathode and an anode. This tube is also known as X-ray tube. The cathode releases electrons due to the effects

of thermal excitation created by the cathode current J. A voltage U accelerates these electrons towards the anode through vacuum. When these electrons hit the anode and release their energy, X-rays are formed [Suetens (2017)]. In modern X-ray imaging machines, cathodes and anodes are primarily made of tungsten [Beutel et al. (2000)]. A diagram of an X-ray tube is illustrated is Figure 4.

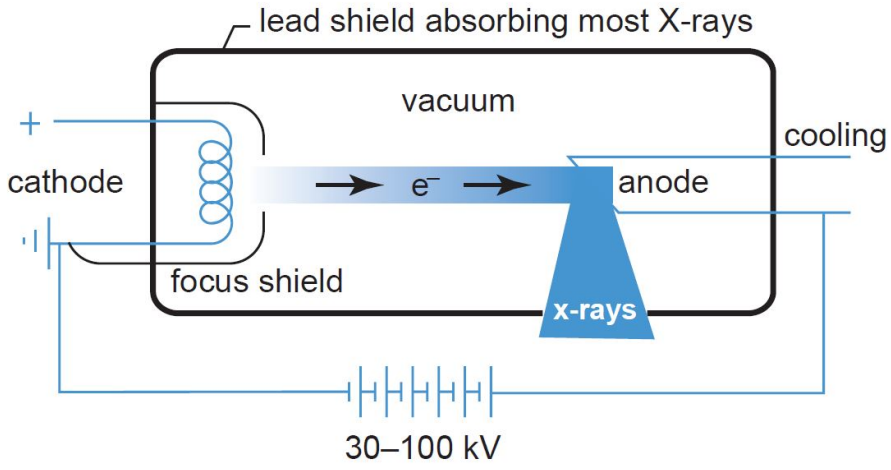


Figure 4: Diagram of an X-ray tube [Suetens (2017)].

Upon interaction with a material, X-rays lose their intensity. The intensity of the outgoing beam I_{out} is related to the intensity of the incoming beam I_{in} by the equation:

$$I_{out} = I_{in}e^{-\mu d}, \tag{2}$$

where d is the thickness of the material and μ is the linear attenuation coefficient which is typically expressed in cm^{-1} [Suetens (2017)]. The value of μ depends upon the energy of the photon and density of the material. Thus, Equation (2) is effective when the density of the material is the same everywhere or the material is homogeneous. In case the material is non-homogeneous, the equation becomes:

$$I_{out} = I_{in}e^{-\int_{x_{in}}^{x_{out}} \mu(x)dx}, \tag{3}$$

where x_{in} and x_{out} are two terminal values of the material.

This attenuated X-ray image is then captured and converted to an image to get information about the density of the material [Suetens (2017)]. X-ray films are one of the oldest but most used form of X-ray detectors. These films are typically made of silver bromide (AgBr) crystals. When these crystals absorb radiation energy they undergo a physical change. The crystal grains absorb the radiation and become dark. The more radiation one area of the film is exposed to, the more crystal grains become dark and that area of the plate becomes darker. Thus, in practice, the X-ray photons when passing through a denser object will lose more of its intensity making part of the film which absorbed those photons brighter [Suetens (2017)]. This is why in X-ray films, bones are brighter than soft tissues or muscles surrounding the bones.

2.4 Fractures of Bones and Indicators of Bone Abnormality

Fractures of bones are common affliction in orthopedic wards of hospitals [Chai et al. (2011)]. Due to the low amount of radiation exposure, X-ray has been employed as a crucial tool in medical imaging, especially in orthopedic diagnosis. Medical professionals such as radiologists examine X-ray images to identify abnormalities such as fractures in bones with a high level of accuracy [Chai et al. (2011)]. Radiologists use measurements of indicators such as radial inclination, and volar tilt to identify the severity of the bone abnormalities [Harisinghani et al. (2018)]. Radial inclination is the angle between line connecting radial styloid tip and and the ulner aspect of distal radius and a second line perpendicular to the longitudinal axis of the radius. Volar tilt is the angle between a line along the distal radial articular surface and the line perpendicular to the longitudinal axis of the radius at the joint. Figure 5 provides visual illustration of these indicators.

Figure 5: Illustration of radial inclination and volar tilt measurement [Harisinghani et al. (2018)].

The range of angle of radial inclination of normal bones is considered to be within 21° and 25° whereas, the range of angle of volar tilt of normal bones is considered to be between 2° and 20° [Harisinghani et al. (2018)]. However, when bones are subjected to abnormal stress, they break or fracture, therefore changing the value of these measurements to be outside the normal range. Based upon the value of these measurements of broken or fractured bones, diagnostic decisions are made and corrective procedures are performed [Harisinghani et al. (2018)]. Thus, along with high quality X-ray images of bones, appropriate measurement of these indicators are paramount in orthopedic diagnosis.

2.5 Deep Feed-Forward Neural Networks

Deep feed-forward networks, otherwise known as feed-forward neural networks or multilayer perceptrons (MLPs), are the quintessential deep learning models [Goodfellow et al. (2016)]. These models aim to approximate some function f^*. In case of classification using deep feed-forward networks, the function $f(x;\theta)$ maps an input x

to a class y while learning the values of parameters θ that are responsible for finding the best approximation of the function [Goodfellow et al. (2016)].

Goodfellow et al. (2016) defined feed-forward networks as functions in a chain like structure with each function in the chain acting as a layer of the network, that is, $f(x) = f^{(3)}(f^{(2)}(f^{(1)}(x)))$. These models are called feed-forward because in these networks information flows in the forward direction, starting from x to the intermediate computations of the functions in the chain and ultimately providing the output y. In the example provided previously, the function $f^{(1)}$ is called the first layer, $f^{(2)}$ is the second layer and so on. Each layer consists of one or many computational units or nodes. These nodes are also called neurons. In this thesis, the nodes in the input layer are not considered as neurons because no computation takes place in those nodes. The depth of the model is described by the overall length of the chain where the final layer is called the output layer [Goodfellow et al. (2016)].

2.5.1 Perceptron

The simplest version of feed-forward neural network also known as perceptron was created by Rosenblatt (1958). A perceptron is made of input units $[x_i]_{i=1}^{D}$, trainable weights $[w_i]_{i=1}^{D}$, a bias w_0, and an output unit y. The structure of the peceptron is given in Figure 6.

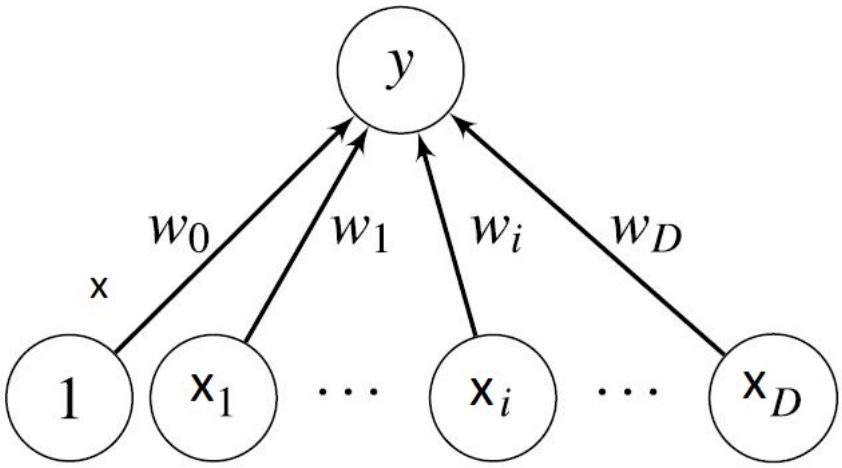


Figure 6: Structure of a single output perceptron.

Since the perceptron has only one layer (not counting the input layer), it is also called a single-layer neural network [Goodfellow et al. (2016)]. In this network, for some input $x \in \mathbb{R}^D$, the output y is obtained by the activation function $f(.)$ which takes a weighted sum of the inputs as follows:

$$y(x;\theta) = f\left(\sum_{i=1}^{D} x_i w_i + w_0\right) = f(w^T x + w_0), \tag{4}$$

where $\theta = \{w, w_0\}$ represents a set of parameters where $w = [w_i]_{i=1}^{D} \in \mathbb{R}^D$ is the set of weights and w_0 is the bias [Zhou et al. (2017)].

When the model has to produce multiple outputs which maps the inputs to different classes, the perceptron described above is not sufficient. Thus, to accomplish this task, multiple output nodes $[y_k]_{k=1}^{K}$ are added to the output layer of the perceptron model [Zhou et al. (2017)]. In this model, each output node y_k represent one class and consist of their respective weights $[w_{ki}]_{i=1;k=1}^{i=D,k=K}$ and biases w_{k0} as follows:

$$y_k(x;\theta) = f\left(\sum_{i=1}^{D} x_i w_{ki} + w_{k0}\right) = f(w_k^T x + w_{k0}). \tag{5}$$

The structure of such a perceptron is given in Figure 7.

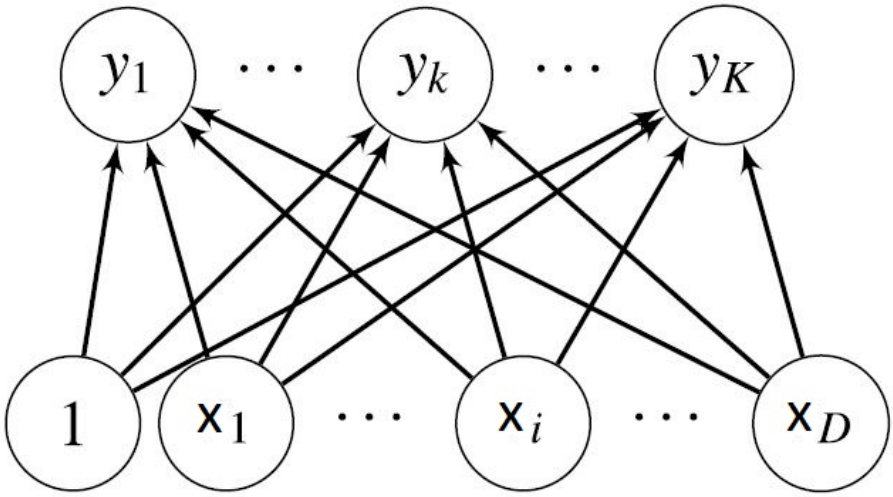


Figure 7: Structure of a perceptron with multiple outputs.

2.5.2 Multilayer Feed-Forward Neural Networks

Two of the main drawbacks of a single-layer neural network are that the model has too few parameters to properly approximate the real output function and the model is incapable of properly separating classes that are not linearly separable [Goodfellow et al. (2016)]. In practical applications, these models are known to have less than optimal performance [Bishop (1995)]. To circumvent the issue of the perceptron model having too few parameters, more layers were added in the middle of input and output layers as intermediate layers which are also known as "hidden" layers [Bishop (1995); Goodfellow et al. (2016)]. To solve the issue of the perceptron model being unable to properly separate classes that are not linearly separable, non-linear activation functions $f(.)$ were added after each additional layer. Figure 8 shows the difference between a pair of linearly separable classes and a pair of non-linearly separable classes. Multiple types of activation functions are discussed in detail in the later part of this section.

The hidden layers are responsible for increasing the number of learnable parameters in the model which can map a set of inputs to a set of output classes [Bishop (1995)].

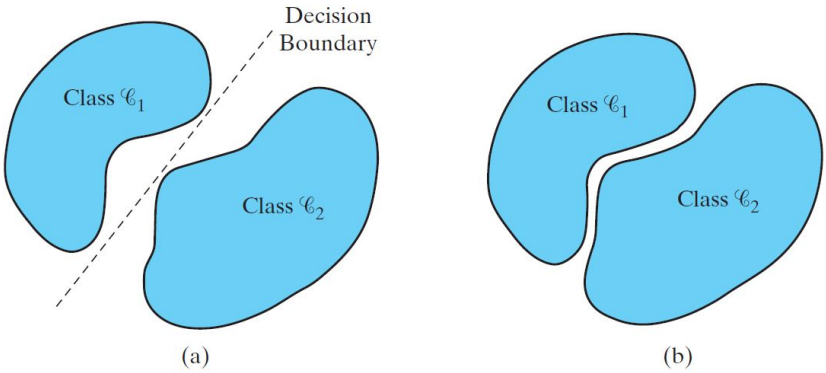


Figure 8: (a) A pair of linearly separable classes. (b) A pair of non-linearly separable classes [Haykin (2009)].

This mapping can be learned by using gradient based methods. The so-called universal approximator property of these models ensures that these models can map any continuous function with a high level of approximation [Hornik et al. (1989)] when non-linearity is applied to the layers. The process of this learning of parameters is called "training" the model [Goodfellow et al. (2016)]. The process of training a model will be discussed in detail in Section 2.6. The composition function of a two-layer neural network which is also known as multilayer perceptron, is given by the equation [Zhou et al. (2017)]:

$$y_k(x;\theta) = f^{(2)}\left(\sum_{j=1}^{M} w_{kj}^{(2)} f^{(1)}\left(\sum_{i=1}^{D} w_{ji}^{(1)} x_i\right)\right). \tag{6}$$

Here, the superscripts denote layer indexes, M denotes number of hidden layers, $\theta = \{w^{(1)} \in \mathbb{R}^{M\times D}, w^{(2)} \in \mathbb{R}^{K\times M}\}$ are the weights, and the bias term is omitted for the sake of simplicity. This composition function can be extended for $(L-1)$ hidden layers as follows [Zhou et al. (2017)]:

$$y_k(x;\theta) = f^{(L)}\left(\sum_{l} w_{kl}^{(L)} f^{(L-1)}\left(\sum_{m} w_{lm}^{(L-1)} f^{(L-2)}\left(\ldots f^{(1)}\left(\sum_{i} w_{ji}^{(1)} x_i\right)\right)\right)\right). \tag{7}$$

During training of these networks, target of these models, whether single-layered or multi-layered is to match the value of y as closely as possible with the original outputs of the data. This can be achieved by tuning the individual parameters in θ during the training of this network [Goodfellow et al. (2016)]. This process will be discussed more in the Section 2.6.

Example of such a feedforward neural network with two hidden layers is given in Figure 9. It should be noted that the layers in the neural network of Figure 9 are "fully-connected layers". Fully-connected layers are the ones where each neuron in the layer is connected to all the neurons in the next layer [Goodfellow et al. (2016)]. However, it is not necessary for a network to be constructed only with fully-connected layers.

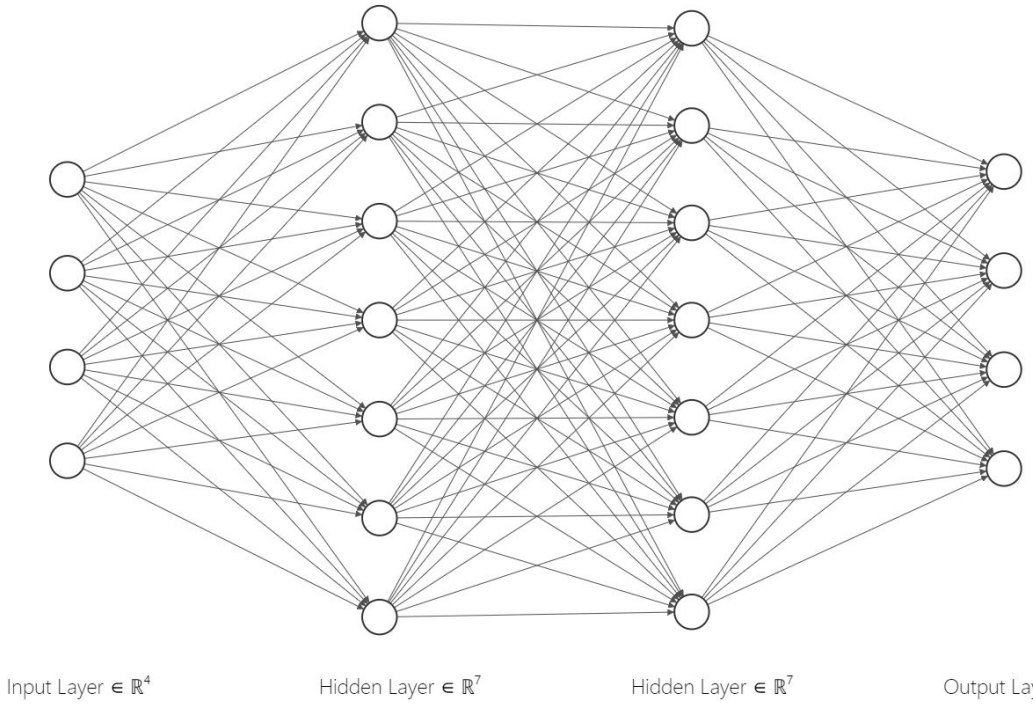


Figure 9: Feedforward neural network with two hidden layers.

Figure 10 illustrates that the model performs better with addition of hidden layers for the task of recognition of multi-digit numbers from photographs of addresses [Goodfellow et al. (2016)]. Performance is judged in terms of accuracy of the model. However, this does not imply that by increasing the depth of the network will ensure perfect accuracy. After the networks reach a certain depth, the accuracy starts to converge [Goodfellow et al. (2016)].

Activation functions are non linear functions that are applied to each neuron of a network to make the model capable of creating complex non linear decision boundaries [Haykin (2009)]. Typically, an activation function limits the output of a neuron. It is also referred to as a squashing function because it squashes (limits) the range of the output of each neurons to a permissible range [Haykin (2009)]. Theoretically, a different activation function can be applied to every layer of a network [Haykin (2009)]. A network with more non-linear layers can approximate non-linear boundaries of the classes of the data better than a network with fewer non-linear layers [Stork et al. (2001)]. Some of the common activation functions include rectified linear unit *(ReLU)* and one different variant of it called *leaky ReLU*, *logistic sigmoid (sigmoid)*, *hyperbolic tangent (tanh)*, and *softplus* [Goodfellow et al. (2016)]. Choice of activation function is very task specific and is often done by trial and error methods to see which function suits the task best however, a recommended choice of activation function for hidden layers is *leaky ReLU* or *ReLU* [Goodfellow et al. (2016)]. The mathematical definitions of the activation functions are as follows:

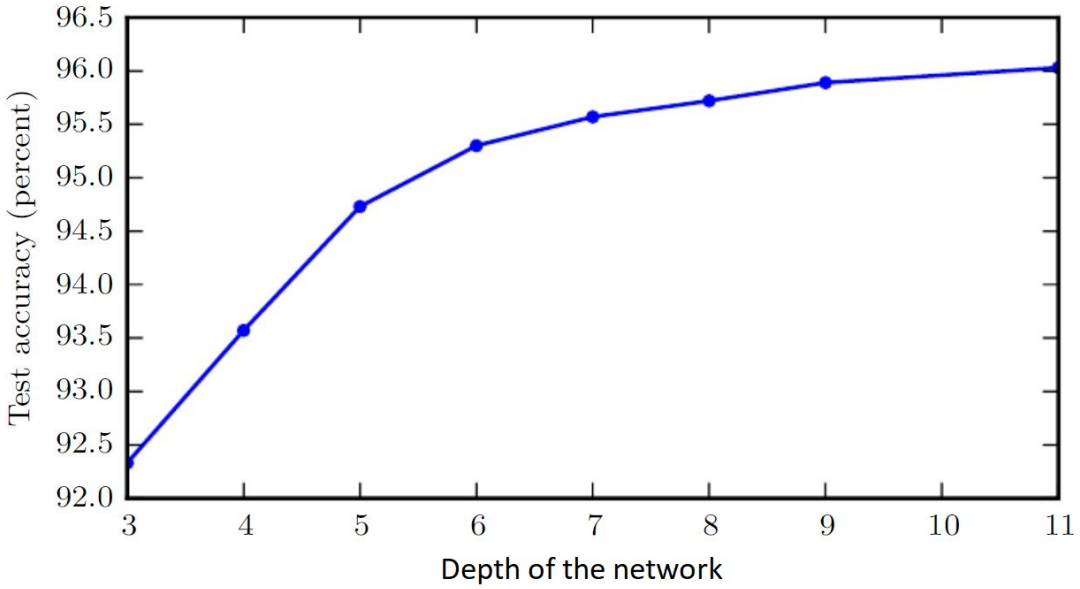


Figure 10: Test Accuracy vs depth of neural network for data described in [Goodfellow et al. (2016)].

$$\text{ReLU}(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{otherwise,} \end{cases}$$

$$\text{leaky ReLU}(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha x & \text{otherwise,} \end{cases}$$

$$\text{sigmoid}(x) = \frac{1}{1 + e^{-x}},$$

$$\tanh(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}},$$

$$\text{softplus}(x) = \log(1 + e^{x}).$$

Visualizations of the activation functions described above along with their derivatives are illustrated in Figure 11. It is evident from the visualizations that there is significant similarity in shape among *ReLU, leaky ReLU, softplus* and also between *sigmoid* and *tanh*. However, they have their differences as well. *ReLU* turns any negative input value to zero where as *softplus* reduces it to zero in a smoother manner. Also, while *softplus* and *ReLU* can not have negative values, *leaky ReLU* keeps a fraction of the negative value of the input function. This fraction is the α parameter. It should be noted that in Figure 11, the value of α is set to be 0.2 for *leaky ReLU*. It can also be observed from the visualization and the equations that the *tanh* function has a range of [-1,1] whereas for *sigmoid* it is [0,1]. [Goodfellow et al. (2016)]

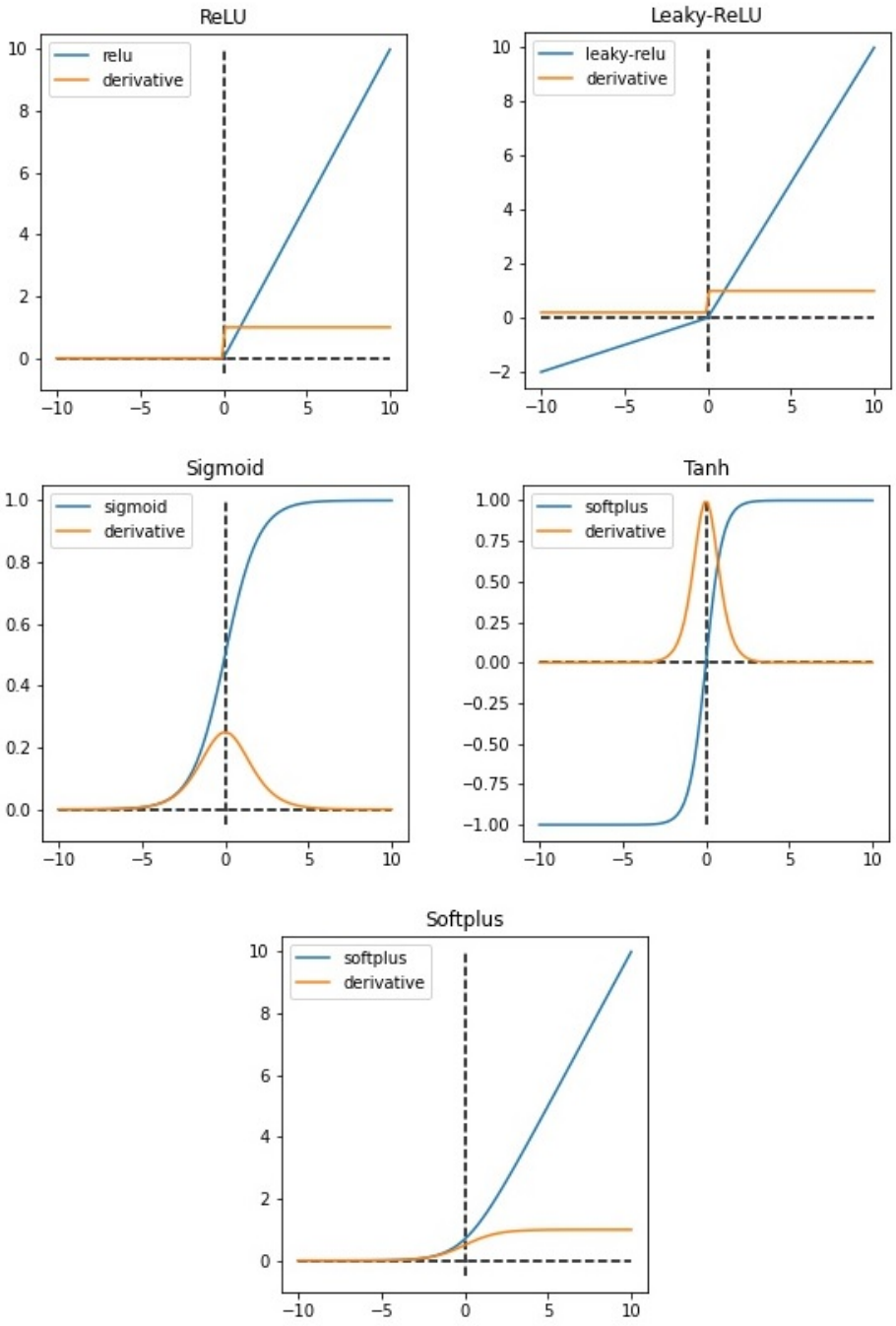


Figure 11: Visualization of the activation functions along with their derivatives.

2.5.3 Types of Deep Neural Networks

Deep neural networks (DNNs) can be categorized into three different types based on what kind of experience they are allowed to take during the process of learning [Goodfellow et al. (2016)]. These categories are supervised learning, unsupervised learning, and reinforcement learning. Algorithms belonging to different categories

adopt different techniques for learning [Goodfellow et al. (2016)]. However, since reinforcement learning is not used in this thesis, it is not discussed in detail.

In supervised algorithms, the model receives training data x containing features and it also receives a label or target value y associated with each instance of training data [Bishop (1995); Goodfellow et al. (2016); Ripley (2007)]. In this case, the model learns to map the input x to its corresponding target value or label y. The labels can be discrete valued where each value represents a class and the model tries to map each input into one or many specific classes. This type of modeling task is called classification [Goodfellow et al. (2016)]. The labels can also be continuous-valued where the task is then to map the input observations into continuous valued labels. This type of modeling task is called regression [Goodfellow et al. (2016)].

In unsupervised algorithms, the model only receives the training data x but does not receive any target value or label y. Therefore, in this case, upon receiving the training data, the model does not try to map the training data to any label. Instead, the model tries to approximate the entire distribution that generated the training data x [Bishop (1995); Goodfellow et al. (2016); Ripley (2007)]. These unsupervised algorithms can also be used for denoising the data or clustering the data where the task is to divide the dataset into clusters of similar observations [Goodfellow et al. (2016)].

The terms "unsupervised" or "supervised" are used depending upon whether the model experiences the labels or not [Bishop (1995); Goodfellow et al. (2016); Ripley (2007)]. If the models do not experience the labels, then it tries to learn probability distribution $p(x)$ or some other properties of the training data. If the model receives the labels y then it tries to predict y from x usually by estimating $p(y|x)$.

2.5.4 Classification Using Deep Neural Networks

Deep neural networks have a very vast range of applications or tasks including classification, regression, clustering, and generative modeling [Goodfellow et al. (2016)]. Classification is the type of task where the aim of the model is to specify which class a particular input observation belongs to. The model produces a function $f : \mathbb{R}^n \to \{1, \ldots, k\}$ to map inputs to a target class [Goodfellow et al. (2016)]. For example, differentiating between image of cats and dogs is a classification task. The output of this classification model is often a conditional probability distribution $p(y|x)$ of the k classes involved, where the output class y depends upon the input observation x [Goodfellow et al. (2016)].

The conditional probabilities of an input belonging to classes are calculated in the output layer of the network [Aggarwal (2018)]. Depending upon number of classes, appropriate activation function can be chosen to calculate the probability of the input belonging to a particular class. If the task is a binary classification problem, that is, if each input is to be mapped into two classes, only one neuron in the output layer is enough to predict the task and we can use the *logistic sigmoid* activation function for it [Aggarwal (2018); Goodfellow et al. (2016)]. The probabilities of x

belonging to a class 0 or 1 can be calculated as:

$$p(y=0|x) = \text{sigmoid}(w^T x + w_0) = \frac{1}{1+\exp[-(w^T x + w_0)]}, \tag{8}$$

$$p(y=1|x) = 1 - p(y=0|x) = 1 - \text{sigmoid}(w^T x + w_0). \tag{9}$$

Here, w is the vector of weights the output layer received from the previous layers and w_0 is the bias in that layer [Aggarwal (2018); Alpaydin (2020)]. The *logistic sigmoid* activation function provides a value ranging between [0,1] which can be considered as the conditional probability of the particular input belonging to class 0, that is, $p(y=0|x)$. Thus, the conditional probability of the same input belonging to class 1 becomes $1 - p(y=0|x)$ [Aggarwal (2018); Alpaydin (2020)].

If the task is a multi-class classification, the output layer will have neurons equal to the number of classes [Aggarwal (2018); Alpaydin (2020); Goodfellow et al. (2016)]. Here, the *softmax* activation function can be used in the neurons of the output layer [Aggarwal (2018); Alpaydin (2020); Goodfellow et al. (2016)]. Output value of the *softmax* activation function of an output node i describes the probability of the input belonging to the class i. The probabilities can be calculated as:

$$p(y=i|x) = \text{softmax}(w_i^T x + w_{0i}) = \frac{\exp(w_i^T x + w_{0i})}{\sum_{j=1}^{k} \exp(w_j^T x + w_{0j})} \qquad \forall i \in \{1, ...k\}, \tag{10}$$

where w_i and w_{0i} denote the weight vector and bias associated with ith neuron in the output layer respectively. It can be seen from the equation that the sum of all the probabilities of all the classes will be equal to 1. After calculating the probabilities of each of the class, the label of the input can be chosen as:

$$\hat{y} = \arg\max_i (p(y=i|x)). \tag{11}$$

Here, the arg max function selects the class with the highest probability value and assigns the class label as the predicted class of the input observation [Goodfellow et al. (2016)].

2.6 Training Deep Feed-Forward Neural Networks with Back-propagation

Section 2.5.2 describe the building blocks of a deep neural network, namely, neurons, weights, biases, and activation functions. The next step is to formulate a method that can change the weights in the network in accordance with the input and desired output. This process of changing weights properly in a feed-forward neural network is referred to learning the parameters of the network [Zhou et al. (2017)]. One technique to learn these parameters is through minimization of an error or cost function [Stork et al. (2001); Goodfellow et al. (2016)]. Let us consider a dataset $[x_n, t_n]_{n=1}^N$ used to train a network, where $x_n \in \mathbb{R}^D$ denotes an instance of training observation and $t_n \in [0,1]^K$ is the indicator of class of the corresponding observation. The class indicator variable is often stored in the form a vector with one-of-K encoding, that

is, only the kth element of the vector will be 1 and remaining elements will be 0 if the observation instance belongs to class k [Zhou et al. (2017)]. One of the most commonly used cost function for K-class classification tasks is cross-entropy [Goodfellow et al. (2016)]. It is defined as follows:

$$E(\theta) = -\sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk}\log(y_{nk}), \tag{12}$$

where t_{nk} represents the kth element of the class indicator vector t_n and y_{nk} represents the kth element of the predicted probability vector y_n for x_n.

To minimize the cost function described in Equation (12), a gradient descent algorithm which updates the parameters in an iterative way can be used. This is done by computing gradient $\nabla E(\theta)$ for the parameters θ [Zhou et al. (2017); Goodfellow et al. (2016)]. In case of a feed-forward neural network, this gradient can be utilized to update the parameters by using gradient backpropagation [Rumelhart et al. (1986)]. Backpropagation algorithm is the way to propagate gradients from output layer of a feed-forward neural network to its input layers by making use of the chain rule of calculus. The derivative of an error function E with respect to the parameters of lth layer $W^{(l)}$ in an L-layer neural network can be obtained by the following equation:

$$\frac{\partial E}{\partial W^{(l)}} = \frac{\partial E}{\partial a^{(L)}}\frac{\partial a^{(L)}}{\partial a^{(L-1)}}\cdots\frac{\partial a^{(l+2)}}{\partial a^{(l+1)}}\frac{\partial a^{(l+1)}}{\partial a^{(l)}}\frac{\partial a^{(l)}}{\partial z^{(l)}}\frac{\partial z^{(l)}}{\partial W^{(l)}}, \tag{13}$$

where $z^{(l)}$ denotes the vector of layer l on which activation function has not been applied yet and $a^{(l)}$ denotes the resulting vector when activation function has been applied on $z^{(l)}$ and $a^{(L)} = y$ [Zhou et al. (2017)]. It should be noted that $\frac{\partial E}{\partial a^{(L)}}$ is the error computed at the output layer and from there the gradient is propagated to the lth layer where $l \in [L-1, L-2, \ldots, 1]$. Once the gradient vector of all the layers are obtained, the set of parameters $W = [W^{(1)}, \ldots, W^{(l)}, \ldots, W^{(L)}]$ is updated as follows:

$$W^{(\tau+1)} = W^{(\tau)} - \eta\nabla E(W^{(\tau)}), \tag{14}$$

where τ and η denote the iteration index and learning rate respectively and $\nabla E(W)$ is obtained via backpropagation as described in Equation (13). The learning rate parameter is responsible for determining how fast the model will reach convergence [LeCun et al. (2012)]. The process of backpropagating gradients and updating parameters is repeated until a desired number of iterations is reached [LeCun et al. (2012)].

There are several approaches to update the parameters as described in Equation (14). One approach is to update parameters based on gradients ∇E calculated over all the training samples. It is called gradient descent. However, this process is very computationally expensive if the number of training samples is very high [LeCun et al. (2012)]. A better way of updating parameters when number of training samples is very high is stochastic gradient descent (SGD) [LeCun et al. (2012)]. In this method, the parameters are sequentially updated by computing gradients on the basis of one sample at a time. This method results in a better generalization of the neural network model. However, in this method, the number of iterations of updating the

parameters increase dramatically. In practice, a trade-off of gradient and stochastic gradient descent method is used. It is called mini-batch stochastic gradient descent [Aggarwal (2018); Li et al. (2014)]. In this method, mini-batches are formed using a small number of training samples and the parameters are updated based on gradients calculated over the mini-batch.

Figure 12 illustrates the difference in updating parameters in stochastic gradient descent and gradient descent. The cross in the center represents the global minima of the loss function. The concentric circles represent the corresponding topographical feature of the loss function. The arrow represents how loss of the model is approaching its global minima. In conventional gradient descent since all of the samples of training data is considered to update parameters, the model proceeds smoothly but in SGD the parameters are updated based on gradients computed by one training sample at a time thus, in this method noise is introduced to the model. This in turn makes the model less smooth so the method takes more iterations to reach the global minima.

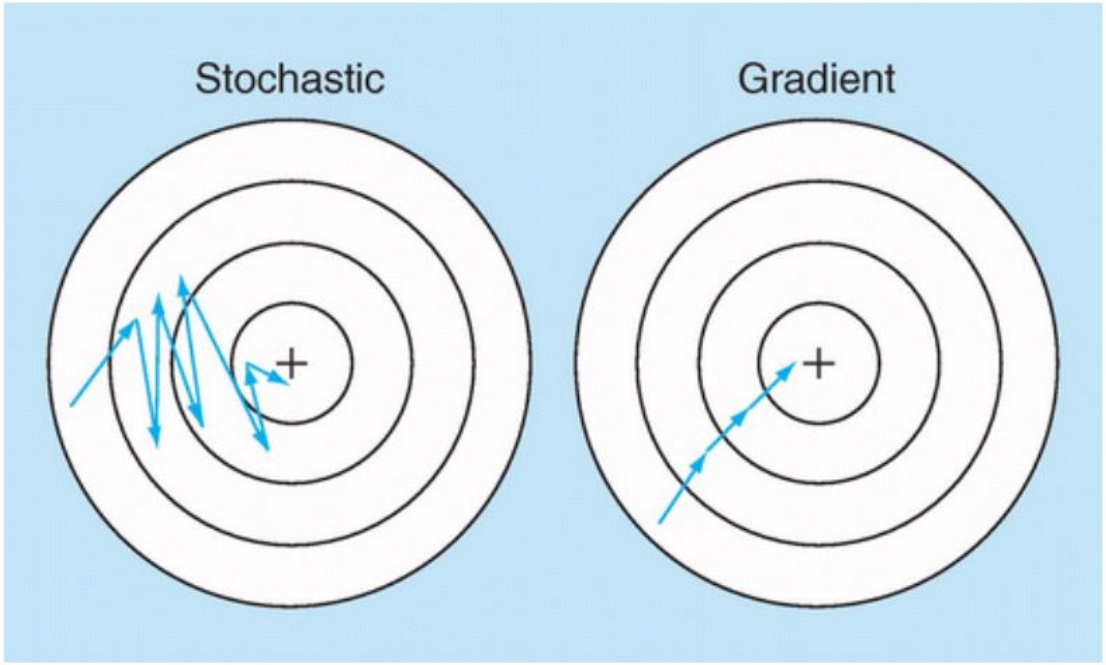


Figure 12: Illustration of difference in updating parameters in stochastic gradient descent and gradient descent [Carpenter et al. (2018)].

One of the more recent parameter updating algorithm developed by Kingma and Ba (2014) is called adaptive moment estimation or Adam. This is the updating algorithm that was used for experimenting during this thesis work. This optimizer computes adaptive learning rates dynamically for each parameter by storing a moving average of mean and variance of the gradients for each parameter [Kingma and Ba (2014)]. The equations used to make these gradient computations are:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t, \tag{15}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2. \tag{16}$$

Here, t and $t-1$ represent the current and previous iterations respectively; β_1 and β_2 are two constants, and m_t and v_t are the moving mean and variance values of the gradient g_t. In their experiments Kingma and Ba (2014) found that the mean and variance values calculated by Equations (15) and (16) are biased towards zero. Thus, they computed bias-corrected mean and variance of the gradients using Equations (17) and (18) respectively. The equations are as follows:

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}, \tag{17}$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t}. \tag{18}$$

Now the bias-corrected mean and variance can be used to update the parameters as:

$$\theta = \theta - \frac{\eta}{\sqrt{\hat{v}_t + \delta}}\hat{m}_t, \tag{19}$$

where δ is a stabilizing constant when $\hat{v}_t$ is near 0.

2.7 Convolutional Neural Networks

Conventional feed-forward neural networks operate on data that is in the form of a vector [Zhou et al. (2017)]. However, if the input data has a grid-like topology (e.g. image), these networks fail to capture spatial information of the data. In case of images, a particular region of the image consisting of a cluster of pixels might hold valuable information. This spatial information gets destroyed when the image is vectorized. To overcome this limitation, convolutional neural networks (CNNs) were developed that specialize in processing data with a grid-like topology [Goodfellow et al. (2016)]. Each layer of the convolutional neural network has 3-dimensional structure [Aggarwal (2018)]. The depth of these layers correspond to the number of features that layer holds. It should be noted that the notion of depth in layers of convolutional neural network is different from the notion of depth in conventional feed-forward neural networks. For example, if color images are inputs of the convolutional neural networks, the input layer will have depth of 3 representing each color channel red, green, and blue and if grayscale images are the inputs then the depth of the input layer will be 1. In essence, one layer of a CNN is stacked with 2-dimensional grids forming a 3-dimensional layer [Goodfellow et al. (2016)].

Convolutional neural networks make use of extensive weight-sharing among its layers which enables these networks to detect features that are invariant to translation [Goodfellow et al. (2016)]. This property helps the network with data in the form of images since relevant features present in the image can reside in multiple spatial locations of the image and even if the image gets shifted in layers, the spatial information remains intact. A convolutional neural network is typically made of three different types of layers: convolution layer, pooling layer, and fully-connected layer. Unlike traditional feed-forward neural networks, the operations in each of these layers are spatially organized and the connections between these layers are sparse [Aggarwal (2018)]. These layers are discussed as follows.

A convolution layer performs a convolution operation which is similar to vector dot product operation on the inputs making the model more effective to detect local patterns present at different parts of the input feature map [Aggarwal (2018)]. It does so by using learnable filters $k_{ij}^{(l)}$. These filters are connection weights between feature map i at the layer $l-1$ and feature map j at the layer l. The features of a convolution layer l compute their activations $A_j^{(l)}$ by performing the convolution operation on the filters $k_{ij}^{(l)}$ and a spatially contiguous subset of units in the feature maps $A_i^{(l-1)}$ of the preceding layer $l-1$ as follows [Zhou et al. (2017)]:

$$A_j^{(l)} = f\left(\sum_{(i=1)}^{M^{(l-1)}} A_i^{(l-1)} * k_{ij}^{(l)} + b_j^{(l)}\right). \tag{20}$$

Here the convolution operation is denoted by the symbol $*$, $f(.)$ is the non-linear activation function of the layer, $b_j^{(l)}$ is the bias parameter, and $M^{(l-1)}$ denotes the number of feature maps in the layer $l-1$. Figure 13 provides an example of convolution operation between a $7 \times 7 \times 1$ dimensional input and a $3 \times 3 \times 1$ dimensional filter. The highlighted boxes indicate the contiguous values of the input which are convolved with the filter and after the filter has shifted over the entire input map with a stride of one unit, the result of the convolution operation is shown in the box on the top right side of the Figure 13 [Aggarwal (2018)].

After convolution layer, pooling layer is used to subsample the feature maps in the convolution layer [Goodfellow et al. (2016)]. The subsampling is done in a similar manner to that of convolution. A filter of a specific dimension strides on the input feature map with a specified stride value and calculates statistics that create a new feature map. One of the statistics is to retrieve the maximum value present in the region of the filter and add it in the new feature map. This is called max-pooling [Goodfellow et al. (2016); Aggarwal (2018)]. Another method is to calculate the average of the values present in the region of the filter and add that value to the new feature map. This is called average-pooling [Goodfellow et al. (2016); Aggarwal (2018)]. Figure 14 illustrates the idea of max-pooling. Here the input feature map is of dimension 7×7 and the filter is of dimension 3×3. When the filter is shifted with strides of 1 and 2, it creates output feature maps of dimension 5×5 and 3×3 respectively. In both of the cases, the highest value present in the region of the filter in input feature map is passed on to the output filter map.

Fully connected layer performs flattening of the feature map it was provided as an input. This flattening operation creates a vector which can be then passed on to suitable activation functions to generate the probability of the input belonging to a particular class [Aggarwal (2018)]. Figure 15 illustrates the architecture of one of the earliest convolutional neural networks called LeNet-5 [Aggarwal (2018)].

2.7.1 Preventing Overfitting

An overly complex neural network with too many parameters to update often creates a phenomena called "overfitting" [Goodfellow et al. (2016)]. It is the phenomena that when a model considers a very high number of weights and connections associated

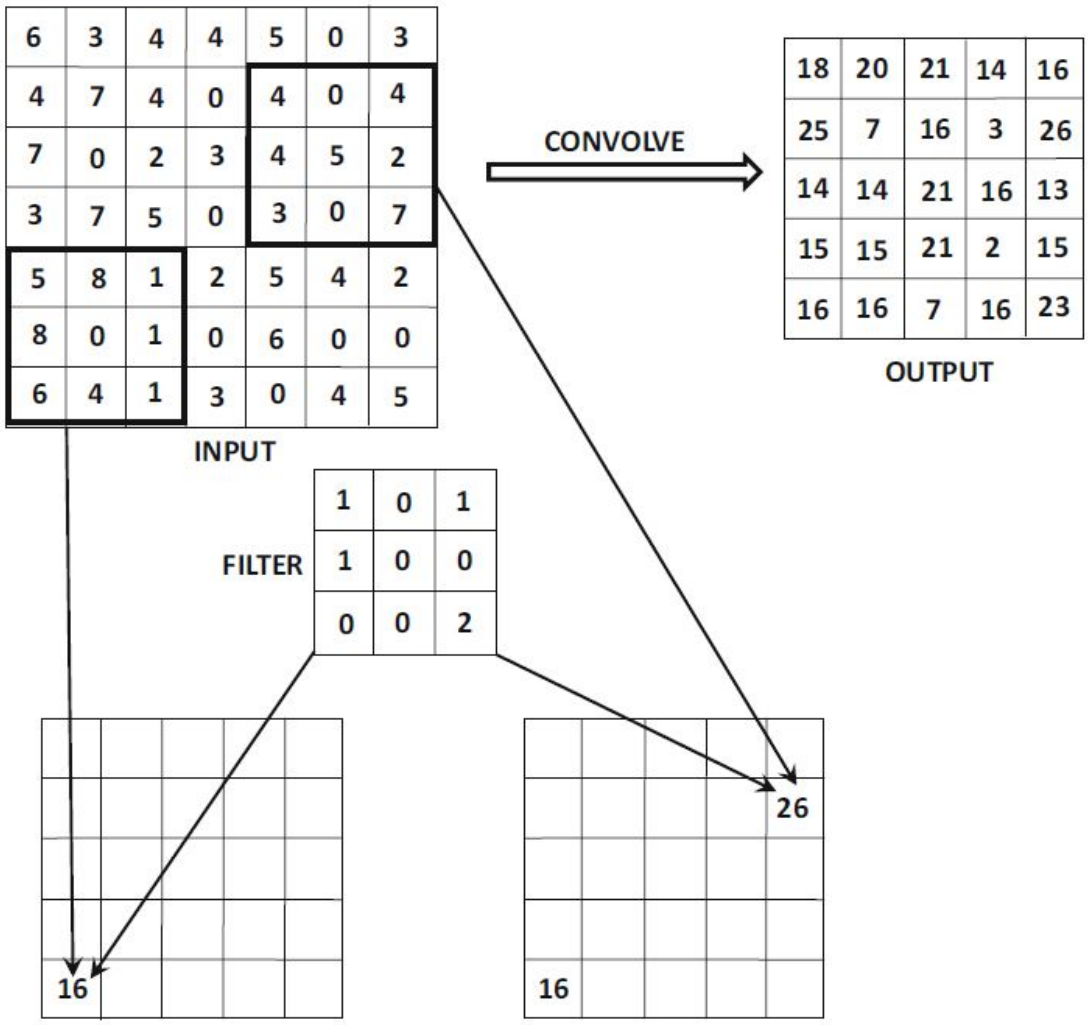


Figure 13: An example of a convolution operation [Aggarwal (2018)].

with each neuron of the network, the model starts remembering specific pattern during its training phase and tries to use those patterns to differentiate among data instances in testing data [Aggarwal (2018)]. This leads to the case where the model performs very well on training data but very poorly on testing data. The high number of neurons and connections also tend to make the creation of model very time consuming. There are many techniques that are used to prevent overfitting but in this thesis only two techniques were used, namely, dropout and data augmentation. Thus, only these two techniques are described in details.

Srivastava et al. (2014) introduced the concept of dropout. They proposed to randomly drop neurons that are not part of the output layer of the base network along with the links associated with it during training. A number is given as a parameter to the dropout function which corresponds to the percent of the neurons that will be dropped out randomly. This idea has been proven to dramatically increase the performance of neural network models [Srivastava et al. (2014)].

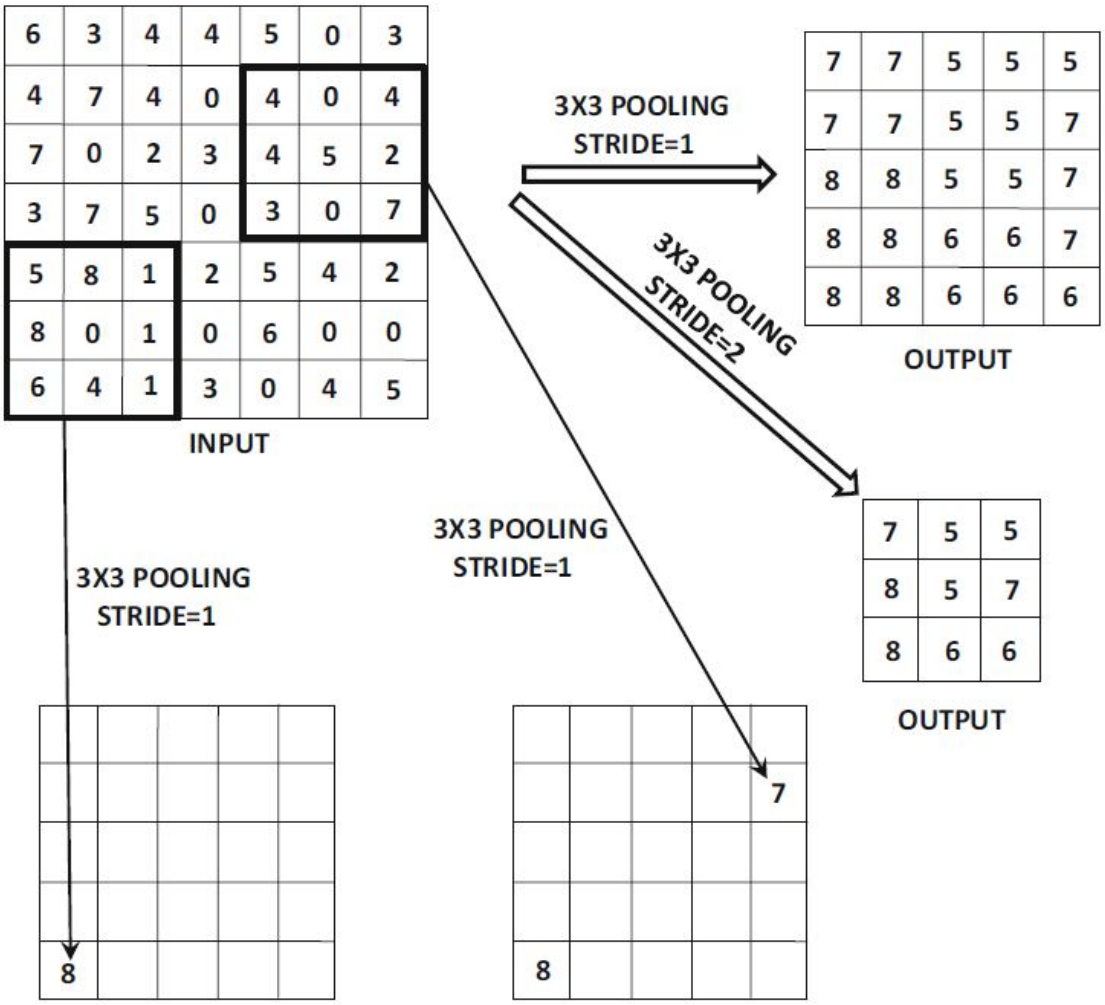


Figure 14: An example of a max-pooling operation [Aggarwal (2018)].

Figure 16 illustrates the results of dropout on a network [Goodfellow et al. (2016)]. On the left, a small network made of two visible neurons, two hidden neurons and one output neuron is considered as the base network. There are 16 possible sub-networks that may be created when dropout is applied to the base network. These sub-networks are shown to the right. It can be seen that many of these sub-networks do not have input nodes or paths connecting input nodes to output nodes. This problem becomes insignificant when the depth and width of the network is significantly large, which is the case for networks created for practical applications [Goodfellow et al. (2016)].

Another popular method to prevent overfitting is data augmentation where data is synthetically created by applying some transformations on the natural data in a supervised manner so that the synthetic data preserves the label of its corresponding natural data [Goodfellow et al. (2016)]. In this thesis the augmentation techniques used for experimentation are rotation, cropping, flipping, and adding synthetically created data using generative modeling.

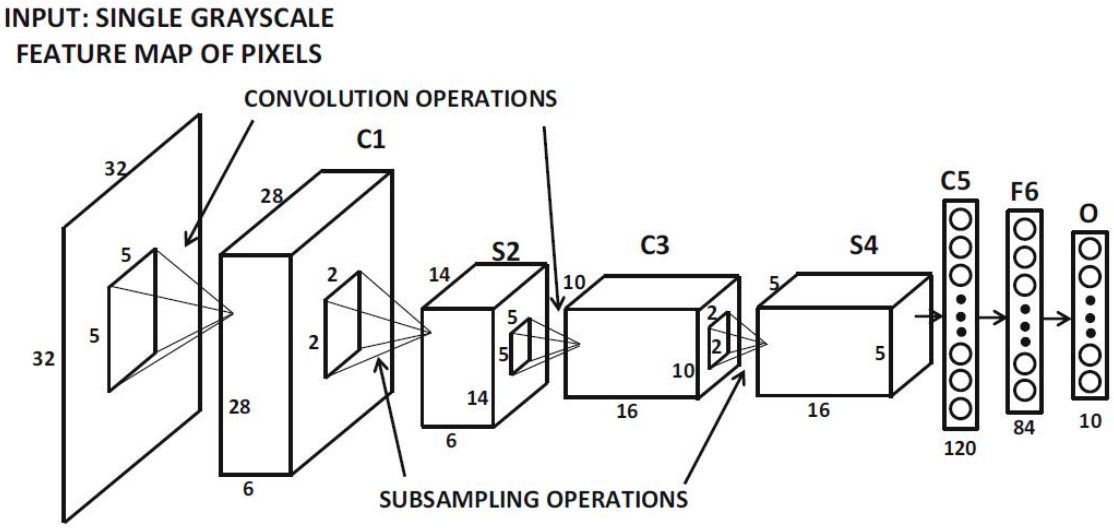


Figure 15: Architecture of LeNet-5 [Aggarwal (2018)].

2.7.2 Deep Residual Networks

One of the main issues of training deep neural networks is that the time for the training process to converge increases dramatically with increase in depth of the networks [Aggarwal (2018)]. Also with increasing depth, the accuracy of the model gets saturated and then starts to decrease dramatically [He et al. (2016)]. However, He et al. (2016) created a model called residual network or ResNet with 152 layers which outperformed every other CNN model of that time and became the first classifier with human-level performance [Aggarwal (2018)]. To create such a deep neural network model, He et al. (2016) proposed the idea of *skip connections*. Skip connections enable copying among layers of the model. While typical neural networks only contain connections between layers i and $(i+1)$, ResNet uses skip connections to establish connections between layers i and $(i+r)$ for $r > 1$ [He et al. (2016)]. Figure 17 illustrates an example of a basic unit of ResNet where skip connection is being used with $r = 2$.

The part of the ResNet shown in Figure 17, consists of two feed-forward layers and one skip connection which simply copies the input of layer i and adds it to the input of layer $i+2$ [He et al. (2016)]. This basic unit is also called a *redisual module* [Aggarwal (2018)] and an entire residual network is built by combining many of these residual modules. This approach increases the speed of updating gradients because in these models, by using the skip connections, gradients can propagate faster in the backwards direction [Aggarwal (2018); He et al. (2016)]. In most of the feed-forward layers of ResNet, appropriate padding filters are used so that the spatial size and depth of the resulting outputs of these layers match that of the input of their corresponding skip connection. Because, if the output of these layers become spatially inconsistent with the input of their corresponding skip connection, the addition operation will not be successfully executed [Aggarwal (2018)].

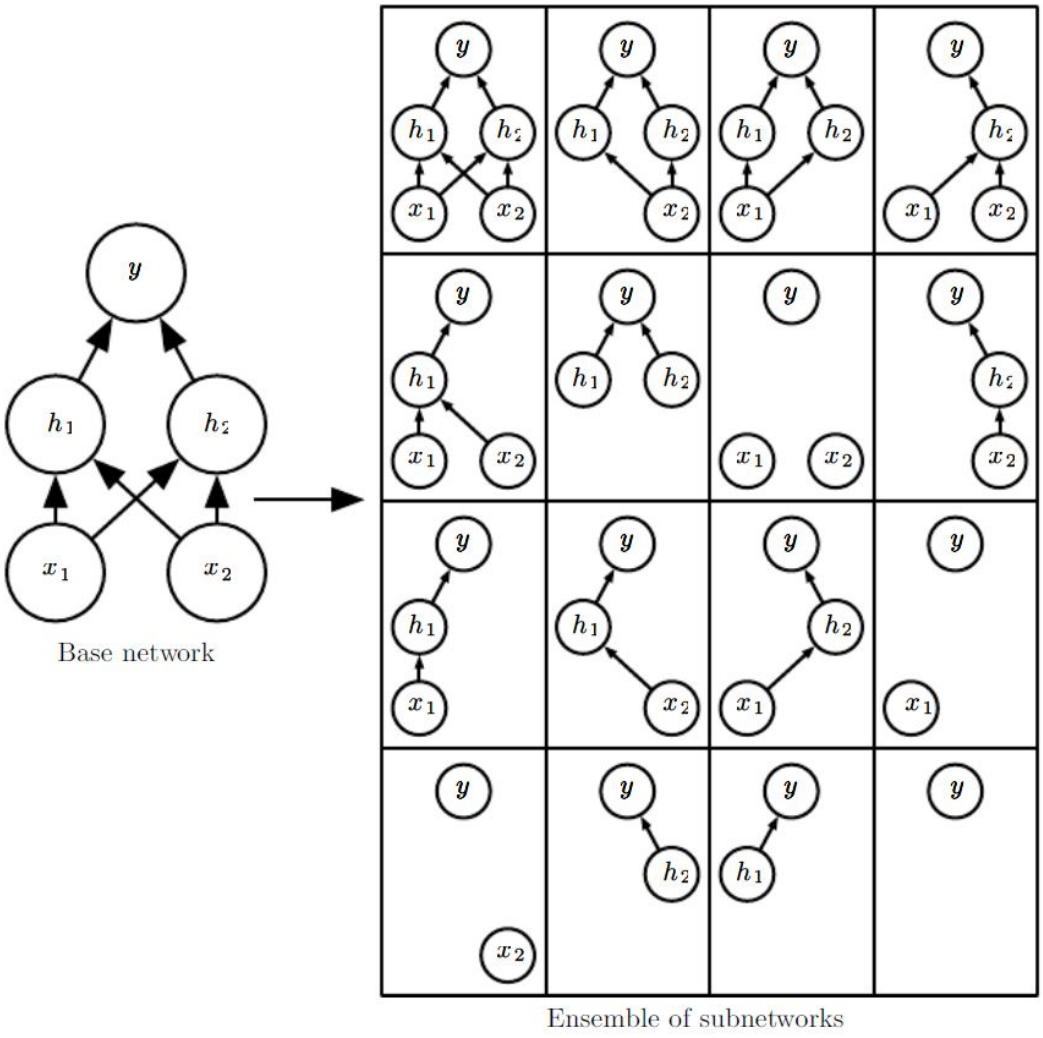


Figure 16: Examples of resulting ensemble of networks when dropout is applied on a simple base network [Goodfellow et al. (2016)].

2.8 Generative Adversarial Networks

Generative adversarial networks (GANs) work with two neural network models in an adversarial game theoretic scenario in which first neural network model competes against the second neural network model [Goodfellow et al. (2014)]. The model which is tasked with generating samples from random noise is called the generator and the other model, called the discriminator, is tasked to distinguish between samples drawn from the generator model and samples drawn from the training data. Upon receiving a sample $\bar{X}$, the discriminator calculates the probability value $D(\bar{X})$ representing the probability of the sample belonging to the training data [Goodfellow et al. (2016)].

Once the discriminator identifies the input as a synthetic object created by the generator or as sample from training data, the information is used to calculate two loss values namely generator loss and discriminator loss [Aggarwal (2018); Goodfellow et al. (2016, 2014)]. Then the losses are used by gradient based methods to update

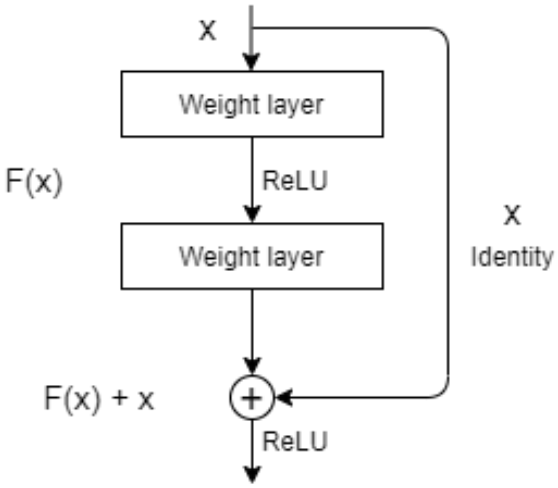


Figure 17: A basic unit of ResNet with one skip connection.

the weights of the generator and discriminator. The weights of the generator are updated in a way that the discriminator will find it more difficult in the next iteration of training to discriminate between data from training set and data created by the generator. This process is repeated and over time the generator learns to produce better counterfeit data. Theoretically, in due course of time, the generator should be able to generate such counterfeits that are impossible for the discriminator to distinguish from samples of training data [Aggarwal (2018)]. Figure 18 illustrates an overview of a GAN model.

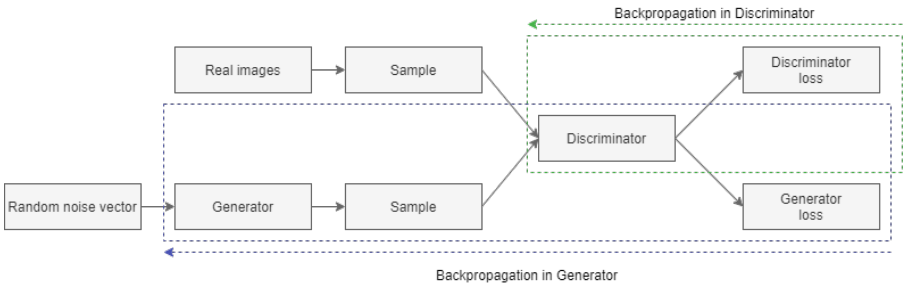


Figure 18: Overview of a GAN architecture.

2.8.1 Training GANs

During training, the discriminator model takes d-dimensional inputs and provides a single output $\in (0, 1)$ which is the probability of the inputs belonging to the training data. The input of the generator model is in the form of noise samples from a p-dimensional probability distribution. And the outputs of the generator are d-dimensional examples of data.

The discriminator aims to correctly classify examples from training data to a label of 1 and data synthetically created by the generator to a label of 0 [Aggarwal

(2018)]. On the other hand, the generator aims to generate examples of data which the discriminator incorrectly classifies to a label of 1. Synthetic samples are generated by creating a set N_m of p-dimensional noise samples $\{\bar{Z}_1, \ldots, \bar{Z}_m\}$ which are fed into the generator to create data samples $S_m = \{G(\bar{Z}_1), \ldots, G(\bar{Z}_m)\}$. Therefore, the maximization objective function J_D for the discriminator is as follows [Aggarwal (2018)]:

$$\begin{aligned} \text{Maximize}_{\text{D}}\ J_D &= \sum_{X \in R_m} \log[D(X)] + \sum_{\bar{X} \in S_m} \log[1 - D(\bar{X})] \\ &= \sum_{X \in R_m} \log[D(X)] + \sum_{\bar{Z} \in N_m} \log[1 - D(G(\bar{Z}))], \end{aligned} \tag{21}$$

where R_m denotes m randomly sampled examples from the training data and S_m denotes m synthetic samples that are generated by the generator. The maximization objective function J_D will be maximized when the examples from training data are classified as 1 and the synthetic examples created by the generator are classified as 0 [Aggarwal (2018)].

Since the generator aims to create samples that the discriminator classifies as training data, the generator objective function J_G, minimizes the likelihood that the generated samples are classified as synthetic. The minimization objective function is as follows [Aggarwal (2018)]:

$$\text{Minimize}_{\text{G}}\ J_G = \sum_{\bar{X} \in S_m} \log[1 - D(\bar{X})] = \sum_{\bar{Z} \in N_m} \log[1 - D(G(\bar{Z}))]. \tag{22}$$

It can be seen that the objective function will be minimized when the synthetic data examples are incorrectly classified as 1. Equation (22) also illustrates that the generator does not need to experience data from the training set to counterfeit it [Aggarwal (2018)]. From Equation (21) and (22) it can be observed that the optimization problem of GANs can be formulated as a minimax game over J_D and J_G. Thus, the overall optimization problem can be written as:

$$\text{Minimize}_{\text{G}}\ \text{Maximize}_{\text{D}}\ (J_G, J_D). \tag{23}$$

In practice, stochastic gradient ascent is used for updating the parameters of the discriminator and stochastic gradient descent is used for updating the parameters of the generator [Aggarwal (2018); Goodfellow et al. (2016)]. The gradient update steps are alternated between generator and discriminator. However, it is common to update one model more frequently than the other to stabilize the overall GAN model.

Stabilizing a GAN model is a difficult task due to several issues. In the early iterations of the training, the generator produces poor samples which in turn makes the task of classification easier for the discriminator [Aggarwal (2018)]. Thus, in the early stages the value of $D(\bar{X})\ \forall\ \bar{X} \in S_m$ stays close to 0. This implies that the value of the loss function of the generator is close to 0, making the gradient update of the loss function very modest which in turn makes the process of training generator models very time consuming. On the other hand, the parameters of the discriminator

gets updated faster due to the gradient of discriminator loss being large. This often leads to a situation called “mode collapse” [Srivastava et al. (2017)]. Mode collapse is the scenario when the discriminator always classifies the images correctly causing the minimax game to break down and the generator to stop learning. As a result, the generator produces very similar samples repeatedly which in turn, reduces the diversity of the generated samples [Srivastava et al. (2017)].

Even though stabilizing a GAN is a difficult task, it has been observed that by careful selection of parameters and model architecture that, it is possible to generate good quality images from the models [Goodfellow et al. (2016)]. In an attempt to make use of deep convolutional neural networks as generator and discriminator of a GAN model, Radford et al. (2015) designed deep convolutional GAN (DCGAN). Their model was able to perform very well for image synthesis tasks. They also showed that the generated images are very sensitive to the input noise samples. Figure 19 illustrates some examples created by the DCGAN model. Each row represents an example of smooth transition in an image by changing the input noise of the model [Aggarwal (2018); Radford et al. (2015)].

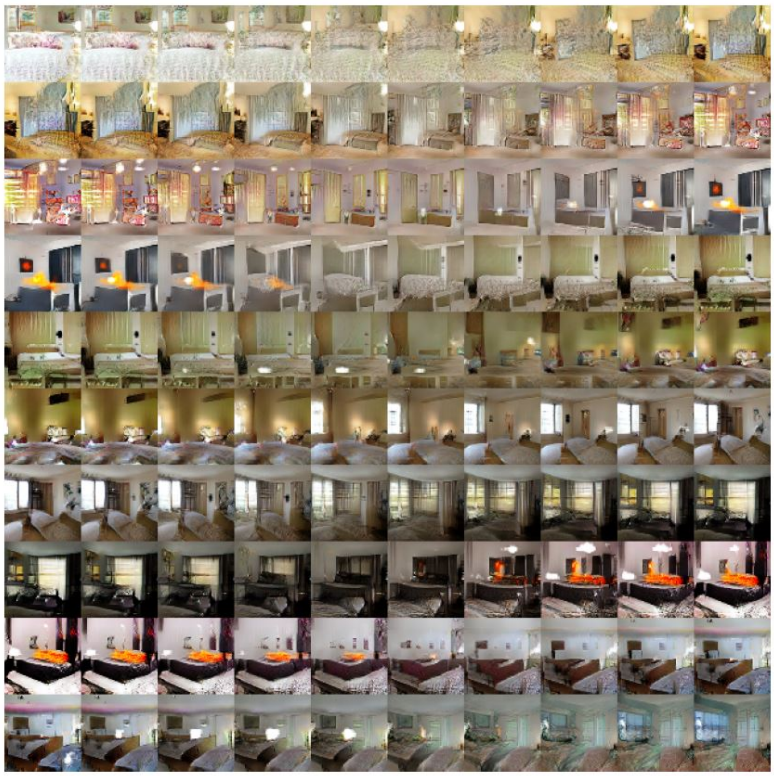


Figure 19: Images of bedrooms generated by the DCGAN model [Radford et al. (2015)].

2.8.2 Conditional GANs

In a typical GAN model, there is no control over what kind of data is being generated. One can only observe the changes in created data by changing the parameters of the model or by changing the input noise samples [Aggarwal (2018)]. However, Mirza and Osindero (2014) introduced a model which is called conditional generative adversarial networks (CGANs), where both the generator and discriminator are created with an added condition as input [Mirza and Osindero (2014)]. The condition can be an additional input object in the form of a class label, a caption, or another object of the same type [Aggarwal (2018)]. Let y denote the condition input to the generator and discriminator. In the generator model, the prior input noise and y are combined in joint representation. And that joint representation is used to train the generator [Mirza and Osindero (2014)]. Thus, for CGANs, Equation (22) can be extended as follows:

$$\text{Minimize}_{\text{G}}\ J_G = \sum_{\bar{Z} \in N_m} \log[1 - D(G(\bar{Z}, y))]. \tag{24}$$

In the discriminator, both X and y are given as inputs to the discriminative function [Mirza and Osindero (2014)]. Similarly, Equation (21) can be extended as follows:

$$\text{Maximize}_{\text{D}}\ J_D = \sum_{X \in R_m} \log[D(X, y)] + \sum_{\bar{Z} \in N_m} \log[1 - D(G(\bar{Z}, y))]. \tag{25}$$

Thus, by using Equations (24) and (25) the overall optimization problem can be written as:

$$\text{Minimize}_{\text{G}}\ \text{Maximize}_{\text{D}}\ (J_G, J_D). \tag{26}$$

By using Equations (24), (25), and (26) the CGAN models are trained. Mirza and Osindero (2014) trained a CGAN model on the Modified National Institute of Standards and Technology (MNIST) handwritten digits dataset where they used class labels of the images as conditions. The class labels represent the digit written on the images so the labels are integers from 0 to 9. Figure 20 presents the resulting images created by their model.

2.8.3 Paired Image to Image Translation Using CGANs

Task of a typical GAN model is to convert some random noise vector z to an output data which counterfeits the training data of the model. However, in CGANs, the input noise vector is also associated with a condition parameter y which forces the model to generate specific types of counterfeit data. Isola et al. (2017) proposed a model using the concept of CGANs to convert images of one domain to another. They used the condition parameter as images for the model to observe. The model is then tasked to convert the observed images to images that are counterfeit of the training data. Let, R denote images in the training data which the generator is trying to counterfeit and r denote a sample image of R. The generator of the model is provided with a noise vector z and a conditional image y as inputs and is tasked to

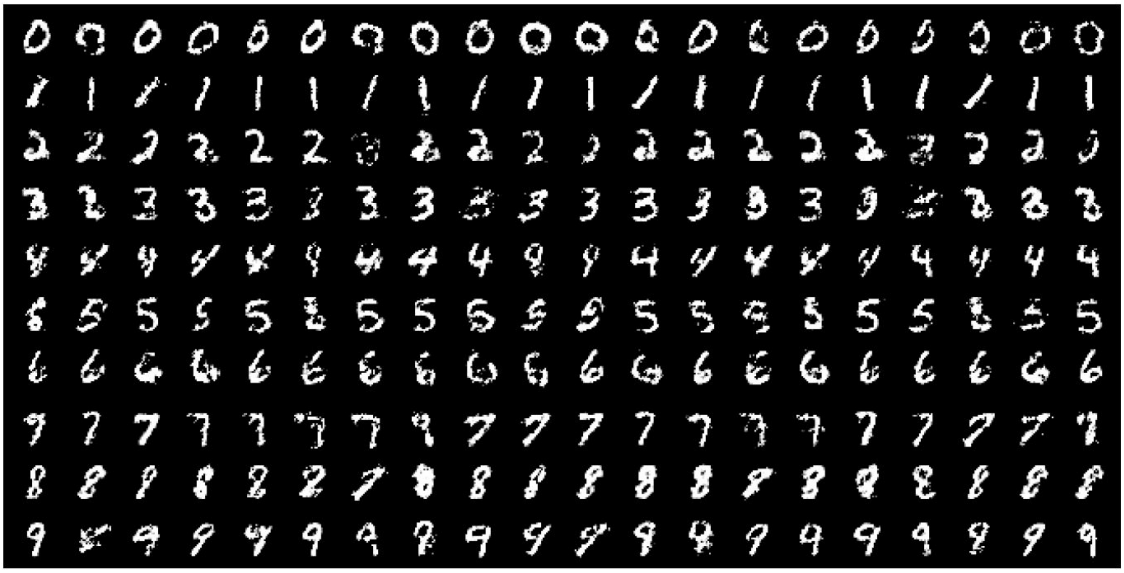


Figure 20: Images of handwritten digits generated by the CGAN model. Images in each row are conditioned with one label and images in each column are different generated samples [Mirza and Osindero (2014)].

convert the image y to an image that counterfeits images in R. This implementation of generator model is typical of CGANs. However, Isola et al. (2017) designed the discriminator model in a way that it takes a pair of images as input. In the pair, one image is the observational image y and the other image is either one that is created by the the generator or an image from R. The training process of this model is illustrated in Figure 21.

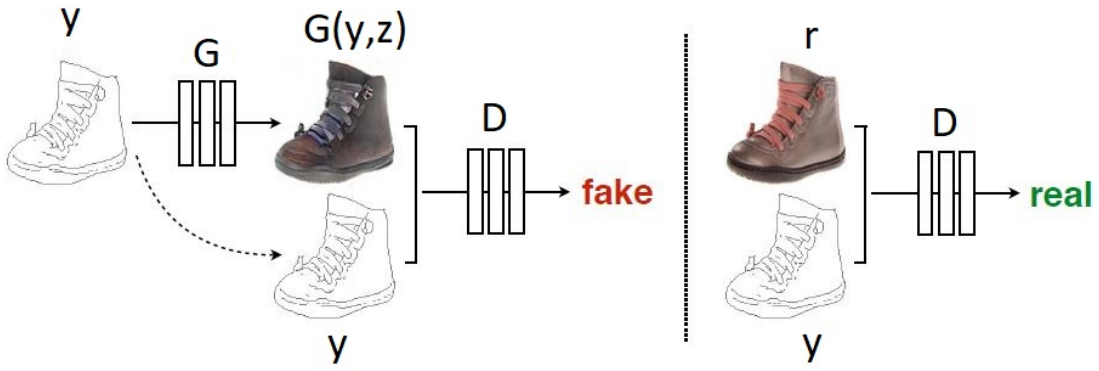


Figure 21: The CGAN converts images of edges to actual pictures. The generator receives noise vector z and edge y and creates a counterfeit image. The discriminator learns to differentiate between the tuple of {edge, counterfeit image} and {edge, original image}, that is, $\{y, G(y, z)\}$ and $\{y, r\}$ [Isola et al. (2017)].

Thus, in this case the minimization objective function of the generator remains the same as the one described in Equation (24). However, Isola et al. (2017) also

added an L_1 distance as a minimization objective. The objective can be expressed as:

$$L_1 = \sum_{X \in R_m, \bar{Z} \in N_m} ||X - G(\bar{Z}, y)||_1. \tag{27}$$

Thus, the modified minimization objective of the generator becomes:

$$\text{Minimize}_\text{G}\ J_G = \sum_{\bar{Z} \in N_m} \log[1 - D(G(\bar{Z}, y))] + \lambda L_1, \tag{28}$$

where λ is a parameter selected manually through experiments. Now, by expanding Equation (25) the maximization objective of the discriminator for this model can now be expressed as:

$$\text{Maximize}_\text{D}\ J_D = \sum_{X \in R_m} \log[D(X, y)] + \sum_{\bar{Z} \in N_m} \log[1 - D(y, G(\bar{Z}, y))]. \tag{29}$$

By using Equations (28), and (29) the final objective becomes:

$$\text{Minimize}_\text{G}\ \text{Maximize}_\text{D}\ (J_G, J_D). \tag{30}$$

Figure 22 Illustrates the results obtained by Isola et al. (2017). In the Figure, the first column represents samples of condition input images, the second column represents samples of real images, third column represents generated images when only L_1 was used as the objective function, fourth column represents generated images when only CGAN objective function was used, and fifth column represents generated images when a linear combination of L_1 and modified CGAN objective function was used. It can be observed that the resulting images in third, fourth, and fifth columns are counterfeits of real images and the overall structure of the counterfeits are similar to that of their corresponding condition inputs.

2.8.4 Unpaired Image to Image Translation Using Cycle-Consistent GANs

The method described in Section 2.8.3 for image to image translation while produced good results, but is also associated with one drawback. To translate an image from one domain to another using the method of Isola et al. (2017), there must be a correspondence between the pair of images of both domains which are fed into the model. However, obtaining paired data for training model can be difficult and expensive [Zhu et al. (2017)]. Thus, to solve this drawback Zhu et al. (2017) designed a model called cycle-consistent generative adversarial network (cycleGAN), which can translate images from one domain to another without providing any paired input to the model. The goal of this model is to approximate mapping functions between two domains X and Y for training samples $\{x_i\}_{i=1}^N$ and $\{y_i\}_{i=1}^N$ where $x_i \in X$ and $y_i \in Y$. Since the aim of this model is to translate images from one domain to another, for two domains, the model needs two mapping functions. One that maps images of domain X to Y which is denoted by $G : X \to Y$ and another that maps images of domain Y to X which is denoted by $F : Y \to X$. To train these mapping functions, two discriminators are also needed which are denoted by D_X and D_Y. The objective

Figure 22: Images generated by the paired image to image translation model [Isola et al. (2017)].

of the discriminator D_X is to distinguish between images $\{x\}$ and translated images $\{F(y)\}$ and similarly, D_Y distinguishes between images $\{y\}$ and $\{G(x)\}$ [Zhu et al. (2017)]. Three different types of objective or loss functions were used to accomplish this task.

2.8.4.1 Adversarial loss

Adversarial loss is the typical loss function of a generative adversarial network which is given by Equations (21), (22) and (23). However, in this case the loss function is applied to both the mappings separately [Zhu et al. (2017)]. The objective for the mapping $G : X \rightarrow Y$ and the associated discriminator D_Y can be expressed as:

$$\mathcal{L}_{GAN}(G, D_Y, X, Y) = \sum_{y \in Y} \log[D_Y(y)] + \sum_{x \in X} \log[1 - D_Y(G(x))]. \tag{31}$$

Here, G aims to minimize the objective and D_Y aims to maximize it, which can also be expressed as $\text{Minimize}_{\text{G}}\ \text{Maximize}_{\text{D}_\text{Y}}\ \mathcal{L}_{GAN}(G, D_Y, X, Y)$. Similarly, for the mapping $F : Y \rightarrow X$ and the associated discriminator D_X the objective can be expressed as:

$$\mathcal{L}_{GAN}(F, D_X, Y, X) = \sum_{x \in X} \log[D_X(x)] + \sum_{y \in Y} \log[1 - D_X(F(y))], \tag{32}$$

where F aims to minimize the objective and D_X aims to maximize it, that is, $\text{Minimize}_{\text{F}}\ \text{Maximize}_{\text{D}_\text{X}}\ \mathcal{L}_{GAN}(F, D_X, Y, X)$.

2.8.4.2 Cycle consistency loss

As explained in Section 2.8.2, images created by adversarial loss are very hard to control. Thus, using only adversarial loss, it will not be possible to guarantee that an input image x_i will be mapped to desired output image y_i. However, instead of making use of CGANs, Zhu et al. (2017) developed a loss function called cycle consistency loss to increase controllability of the model. They argued that to properly map images from one domain to another, the learned mapping functions should be cycle consistent, that is, an image translation cycle should be able to bring some input x back to the original image. Cycle consistency can be expressed as $x \rightarrow G(x) \rightarrow F(G(x)) \approx x$. This property is called *forward cycle consistency*. Similarly, it can also be argued that $y \rightarrow F(y) \rightarrow G(F(y)) \approx y$. This property is called *backward cycle consistency*. Cycle consistency loss can be expressed by combining both of these properties as follows:

$$\mathcal{L}_{cyc}(G, F) = \sum_{x \in X} ||F(G(x)) - x||_1 + \sum_{y \in Y} ||G(F(y)) - y||_1. \tag{33}$$

Equation (33) calculates sum of L_1 distances between the images and their reconstructed versions created by passing the image through one cycle. By minimizing this loss function, the model receives an incentive to create images that remain cycle consistent.

2.8.4.3 Identity loss

By experimenting, Zhu et al. (2017) found that to preserve color composition between the input and output, use of another additional loss function is helpful. This loss function is called *identity loss*. The *identity loss* incentivizes the model to preserve the properties $x \rightarrow F(x) \approx x$ and $y \rightarrow G(y) \approx y$. By combining both these properties, identity loss can be expressed as:

$$\mathcal{L}_{identity}(G, F) = \sum_{x \in X} ||F(x) - x||_1 + \sum_{y \in Y} ||G(y) - y||_1. \tag{34}$$

By combining the loss functions described in Equations (31), (32), (33), and (34) the full objective of the cycle-consistent GAN can be formulated. The full objective can be expressed as [Zhu et al. (2017)]:

$$\begin{aligned}\mathcal{L}(G, F, D_X, D_Y) = \mathcal{L}_{GAN}(G, D_Y, X, Y) + \mathcal{L}_{GAN}(F, D_X, Y, X) + \lambda \mathcal{L}_{cyc}(G, F) \\ + \gamma \mathcal{L}_{identity}(G, F),\end{aligned} \tag{35}$$

where λ and γ parameters represent the relative importance of the losses $\mathcal{L}_{cyc}$ $\mathcal{L}_{identity}$ respectively [Zhu et al. (2017)]. These two parameters are chosen through experiments. Figure 23 illustrates some samples of results generated by Zhu et al. (2017). Images in the top two rows of the Figure 23 demonstrate results of image translation between horse and zebra, the middle two rows are results of image translation between seasons summer and winter, and the bottom two rows are results of image translations between apples and oranges [Zhu et al. (2017)].

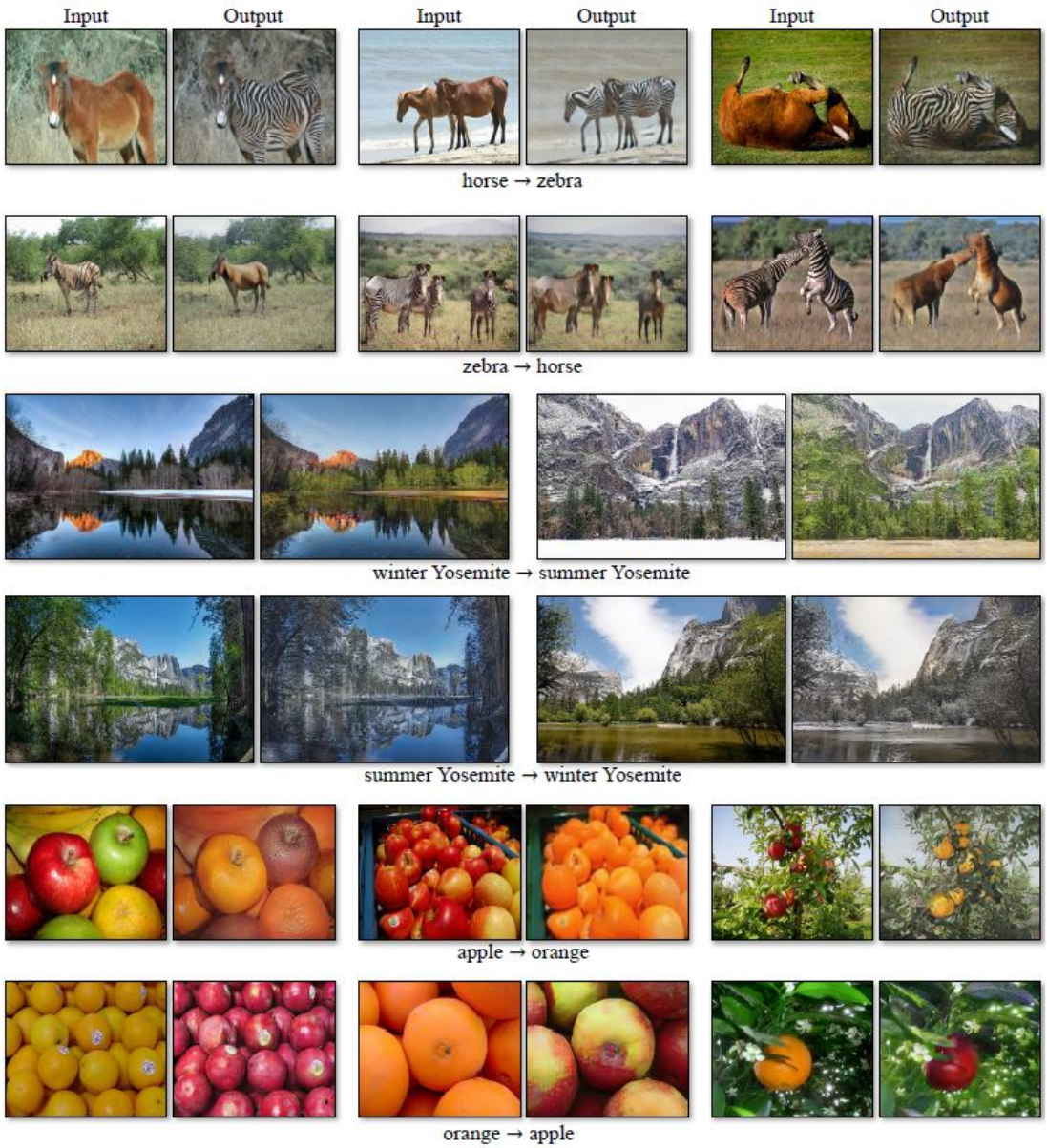


Figure 23: Images generated by the unpaired image to image translation model [Zhu et al. (2017)].

2.9 Literature Review of Synthetic Medical Image Generation

Use of cycleGANs have been used in several studies for data augmentation [Zhu et al. (2018), Liu et al. (2018), Perez and Wang (2017)]. Zhu et al. (2018) used cycleGAN to improve the performance of classifying emotions from images. They trained cycleGAN models to generate synthetic images with 6 different emotions namely, fear, angry, disgust, sad, happy, and surprise. Then the generated images were added to appropriate classes of several emotion recognition databases such as Facial Expression Recognition Database (FER2013), Static Facial Expressions in the

Wild (SFEW) database and Japanese Female Facial Expression (JAFFE) database. By adding the synthetically created images of different emotions, they observed that the performance of the classifier has been increased by 5-10%.

In their paper, Liu et al. (2018) discussed methods to detect brain slices in microscopic images. For the task of brain slice detection, they used object detection algorithms such as single shot multibox detector (SSD) [Liu et al. (2016)], you-only-look-once (YOLO) version 3 [Redmon et al. (2016)] and R-CNN [Girshick et al. (2014)]. Due to lack of training data, they used cycleGAN to generate synthetic microscopic images of brain slices. They translated original microscopic images into several artistic style domains such as Monet, Van Gogh, Cezanne, and Ukiyo-e. Then the translated images along with the original images were passed to the same single shot multibox detector model. They observed that the performance of the object detection model was increased significantly for every object detection algorithm.

Perez and Wang (2017) explored effects of data augmentation using cycleGAN for classification purposes. However, their approach to the experiments was different. Instead of using deep neural networks on small datasets, they used a small neural network with only 3 convolutional layers on small dataset to observe the results. They used tiny-imagenet-200 dataset for classification which has 100,000 images of 200 different for training. Each class only had 500 images to train the classifier. They translated the images of the dataset to different themes such as night/day, winter/summer and used the translated images for augmenting the dataset to make it bigger. Then the augmented dataset was used for classification. The results obtained by their experiments show that by using cycleGAN to augment data, performance of the classifier increased around 10% on testing dataset.

Fuhl et al. (2019) used cycleGAN to segment pupils from images of eyes. They used cycleGAN in paired setting. Each pair consisted of grayscale images of the eye and the same image segmented. In the segmented images, the surrounding part of the eye was colored red, the pupil of the eye was colored green and the rest of the visible part of the eye was colored blue. After training the model, they passed grayscale images of the eye to evaluate the quality of segmentation of the images and observed that the cycleGAN model was able to properly segment all 3 parts of the images and applied same color scheme to those parts as well. The results observed by them are presented in the Figure 24.

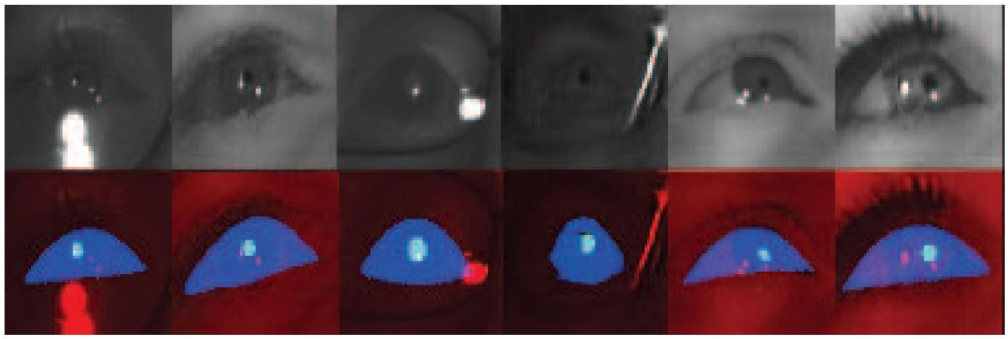


Figure 24: Samples of segmented images generated by Fuhl et al. (2019).

Sandfort et al. (2019) used cycleGAN to translate contrast-enhanced computed tomography (CT) scans to non-contrast CT scans. Then used the translated non-contrast CT scans to augment a dataset consisting only of contrast-enhanced CT scans which is used for organ segmentation. They argued that organ segmentation from CT scans are universally done using contrast-enhanced CT scans but, a large proportion of CT scans are done in non-contrast settings. So, to make the organ segmentation dataset more realistic, non-contrast CT scan images must be added with contrast-enhanced CT scan images. To segment organs from CT scan images, U-Net [Ronneberger et al. (2015)] was used by them. Organ segmented images by experts on contrast-enhanced CT scans were used as masks for the U-Net model. They observed that by augmenting the original images with translated non-contrast CT images produced better segmentation results as compared to standard augmentation techniques such as flipping, rotating and cropping. Examples of translated images are shown in Figure 25. CT scan samples on the left are examples of contrast-enhanced CT scan images and samples on the right are their corresponding translated non-contrast CT scan images.

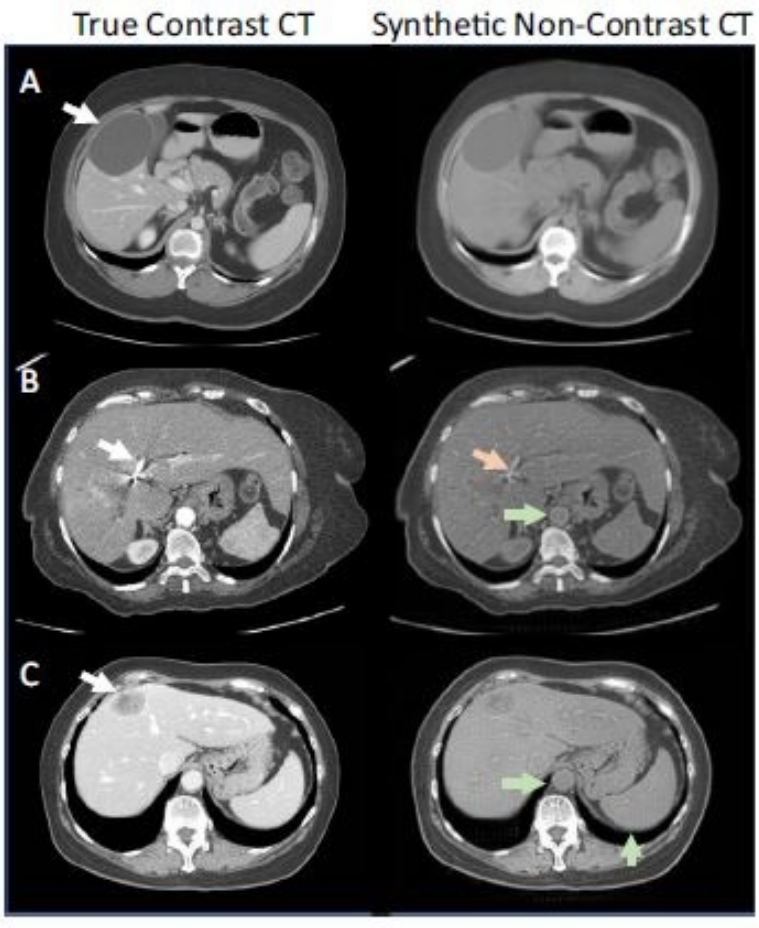


Figure 25: Samples of translated images generated by Sandfort et al. (2019).

Figure 26 Illustrates the segmentation results of the U-Net model. Original CT scan images and organ segmentation done by experts are shown in the first and second

column. The third column shows results of organ segmentation when translated CT scan images were used for data augmentation. And the fourth column represents organ segmentation results where only standard augmentation techniques were used.

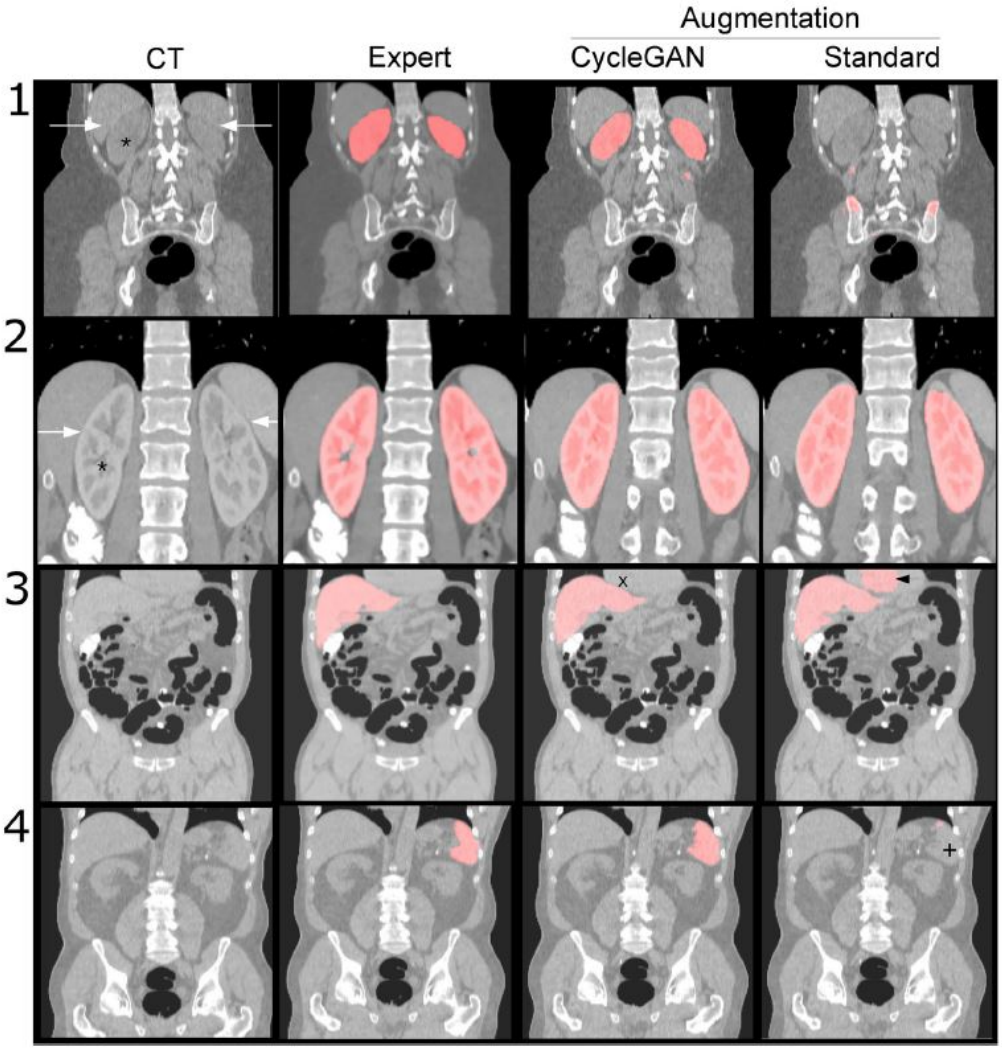


Figure 26: Samples of segmentation results generated by Sandfort et al. (2019).

Use of cycleGAN for image augmentation and segmentation were studied by Zhang et al. (2018); Huo et al. (2018); Seeböck et al. (2019). Zhang et al. (2018) used cycleGAN to translate CT scan images to MRI images and then augmented the translated MRI images to original MRI dataset to improve performance of models that segment organs from MRI images. Huo et al. (2018) translated MRI images to CT images and used the translated CT images to augment dataset consisting CT images for organ segmentation. They observed improvements in segmentation results. Zhang et al. (2018) used CycleGAN on optical coherence tomography (OCT) images. They translated low quality Cirrus OCT B-scan images to higher quality Spectrails OCT B-scan domain to segment retinal fluids from them.

3 Materials and Methods

This section describes a chronological order of the progress of the thesis. In the beginning several papers were studied to explore different methods employed by researchers for data augmentation and image segmentation. Then appropriate datasets were chosen as sources of natural and synthetic X-ray images. In the end, experimentations were performed on the obtained data.

3.1 Datasets Used

For the purpose of experimentation, three datasets were used in this thesis.

3.1.1 MURA Data

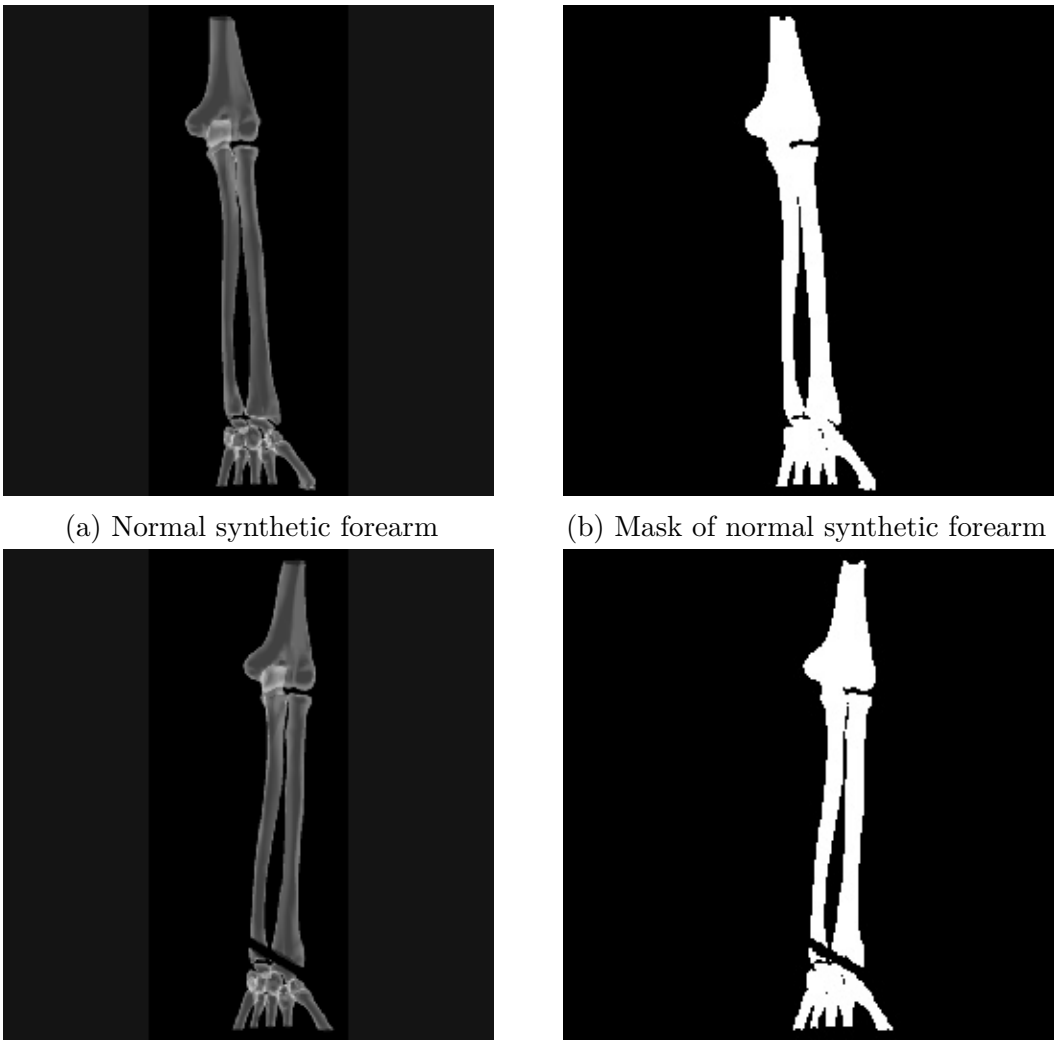
The MURA dataset hosted by the website Kaggle was used as a source of natural x-ray images. Rajpurkar et al. (2017) created the MURA dataset which has a large collection of X-ray images. The dataset contains 40,561 images from 14,863 studies, where each study is manually labeled by radiologists as either normal or abnormal based on the structure of the bones. The dataset consists of images of elbow, finger, forearm, hand, humerus, shoulder, and wrist. Images of each body part has labels normal or abnormal associated to them. Since the task of generating annotations is specific and the types of annotations vary vastly with different body parts, only the forearm images were chosen for experimentation.

The dataset consists of 1,164 normal forearm images for training and 150 normal images for testing as well as 661 abnormal forearm images for training and 151 abnormal forearm images for testing. Sample X-ray images of normal and abnormal forearms in MURA data are shown in Figure 27. In the next chapters of the thesis only the forearm images of the MURA dataset will be called as MURA data.

3.1.2 Mathematica Data

X-ray like images of the forearm and wrists were created using Mathematica module called anatomyplot3d [Wolfram Research (2019)]. The positions of the wrist or forearm in the images were randomly rotated in the range of $\pm 20^{\circ}$ to create variation. Then abnormalities were introduced to half of the Mathematica forearm images by removing parts of bones of the forearm. The abnormalities varied in thickness, position and angle to create variation. No part of bones were removed from the images of wrists as the wrist images were not used for classification purposes. The wrist images were only used to train a model that is able to segment bones from natural X-ray images of wrists, so binarized version of the Mathematica wrist images were also extracted from Mathematica. These binary images served as masks of the synthetic wrist data. Similarly, forearm images were also used to train a model that can segment bones from natural X-ray images of forearms thus, binarized images of the Mathematica forearms were also created using Python which served as masks of the synthetic forearm images. Sample images of the synthetic data are shown in Figure 28 and Figure 29.

Figure 27: Examples of natural X-ray images in MURA dataset [Rajpurkar et al. (2017)]. Images to the left are of normal forearms and images to the right are of abnormal forearms.



(a) Normal synthetic forearm (b) Mask of normal synthetic forearm

(c) Abnormal synthetic forearm (d) Mask of abnormal synthetic forearm

Figure 28: Examples of synthetic images of forearm with their corresponding masks.

3.1.3 RSNA Bone Age Data

The X-ray images of wrists provided by the RSNA bone age dataset which is hosted by Kaggle were used as source of natural X-ray images to train and test a model for image segmentation task [Halabi et al. (2019)]. The dataset has 12,611 X-ray images of wrists for training the model and 200 images for testing the model. Even though the dataset was created to detect age of the person whose hand is in the X-ray image, the data was used to segment bones from the images due to the high quality of the images in the dataset. Two sample images from the dataset are shown in Figure 30.

(a) Synthetic left wrist

(b) Mask of the synthetic left wrist

(c) Synthetic right wrist

(d) Mask of the synthetic right wrist

Figure 29: Examples of synthetic images of wrist with their corresponding masks.

(a) Natural right wrist

(b) Natural left wrist

Figure 30: Samples of X-ray images of natural wrists in the RSNA dataset [Halabi et al. (2019)].

3.2 Experiments

Creating deep learning models was the primary focus of experimentation. These models were created to accomplish three major tasks.

1. Creating a cycle-consistent generative adversarial network (cycleGAN) model to convert synthetic images into refined natural looking images.

2. Creating a classification model to predict the labels of normal and abnormal images before and after adding refined synthetic images to the original MURA dataset.

3. Expanding the cycleGAN network so that it can accommodate more than two domains of transformation to segment bones in natural images of MURA and RSNA data. In this thesis it is called multi-cycleGAN.

Developing these models consisted of four major phases that are common to developing any deep learning model: data acquisition, data preparation, model training, and model testing. All of the models as well as scripts to prepare the data were created using Python programming language. There are multiple frameworks available to implement neural networks using Python. Pytorch [Paszke et al. (2019)], Keras [Chollet et al. (2015)], Tensorflow [Abadi et al. (2015)], CNTK [Seide and Agarwal (2016)], and Theano [Theano Development Team (2016)] are examples of some of the widely used frameworks for deep learning. For this thesis, only Pytorch was used as the framework to implement the deep learning models. Models were trained and tested on Google Colab cloud platform [Bisong (2019)] that provides a variety of graphical processing units (GPUs).

3.2.1 Preparing the Data

The images in the MURA and RSNA datasets had varying resolutions. All the images were resized to a shape of 256×256 so that the images in all three datasets have same resolution. The images of all three datasets also had 3 color channels namely red, green, and blue. However, since those are X-ray images, colors are unnecessary so the images were converted to single channel grayscale images.

In MURA dataset, the images were vastly varying in terms of quality. Some images had very good positioning of arm and the resulting X-ray images were also good but in a lot of the images, it was very hard to differentiate between the bones and the soft tissues surrounding the bones. To avoid this issue, contrast limited adaptive histogram equalization (CLAHE) was performed on the images.

Histogram equalization which is an image enhancement method was used to adjust the contrast of the images [Pizer et al. (1987)]. It calculates the cumulative density function of grayscale levels of an image and uses it to rescale the grayscale values of that image. Contrast limited adaptive histogram equalization is a method that applies the histogram equalization into small regions or grids of an image and moves the grid across the image until it covers the whole image [Reza (2004)]. It also uses a clip limit parameter to control the maximum number of pixels each grayscale

level is allowed to have [Reza (2004)]. For this thesis, the clip limit parameter was set to 3 and the grid size was chosen to be 8×8. Some examples of resulting images after CLAHE activation is shown in Figure 31. It can be noticed that in the resulting CLAHE activated images, it is easier to differentiate between the bones and the soft tissues surrounding it.

Different set of data augmentation techniques were used for different deep learning models. For the cycleGAN and multi-cycleGAN, the augmentation techniques were random horizontal flipping ("left-to-right" flips), and random cropping. For the classification model, the augmentation techniques included, random cropping, random horizontal flipping ("left-to-right" flips), random rotating with an angle range of $\pm 20°$.

3.2.2 Image to Image Translation Using CycleGAN

Figures 27(a) and 27(b) are examples of X-ray images of normal and abnormal natural forearms respectively and Figures 28(a) and 28(c) are examples of X-ray images of normal and abnormal Mathematica generated forearms respectively. It can be observed that a big difference between the natural and Mathematica generated images is absence of soft tissues around the bones in the later ones. Since the goal of creating synthetic data is to use it for augmenting the dataset used for classification of bone defects, the synthetic data had to have close resemblance with natural data. Thus, to transform the Mathematica generated forearm images into synthetic images that have close resemblance with natural data, image to image translation technique was adopted. However, since the natural images in MURA data varied a lot from the Mathematica images, the best method was to learn unpaired image to image translation mappings than can translate images of one domain to another. Thus, for this task, the cycleGAN model described in Section 2.8.4 was used. It should be noted that Mathematica images of wrists were not translated to natural images of wrists in RSNA data because the Mathematica images of wrists were already very similar to the natural wrist images in RSNA data which can be observed in Figures 29(a), 29(c) and 30(a), 30(b).

Let A and B denote the two domains representing forearm X-ray images from MURA and Mathematica respectively. Thus, the required task is to translate images from domain B to domain A. Let, G_{AB} denotes the generator model that translates images from domain B to A and similarly G_{BA} denotes the generator model that translates images from domain A to B. Also, let the two discriminator models be denoted by D_A and D_B. The discriminator D_A aims to distinguish between images $\{a\}$ $\forall a \in A$ and translated images $G_{AB}(b)$ $\forall b \in B$. Similarly, the objective of discriminator D_B is to distinguish between images $\{b\}$ $\forall b \in B$ and translated images $G_{BA}(a)$ $\forall a \in A$.

The adversarial losses for this model were calculated in a different way than the one described in (31) and (32). The least square GAN loss described by Mao et al. (2017) was used in this model for adversarial loss. Instead of maximizing the loss of the discriminator, the loss functions of discriminators in this model were also minimized alongside the loss functions of the generators. The adversarial loss

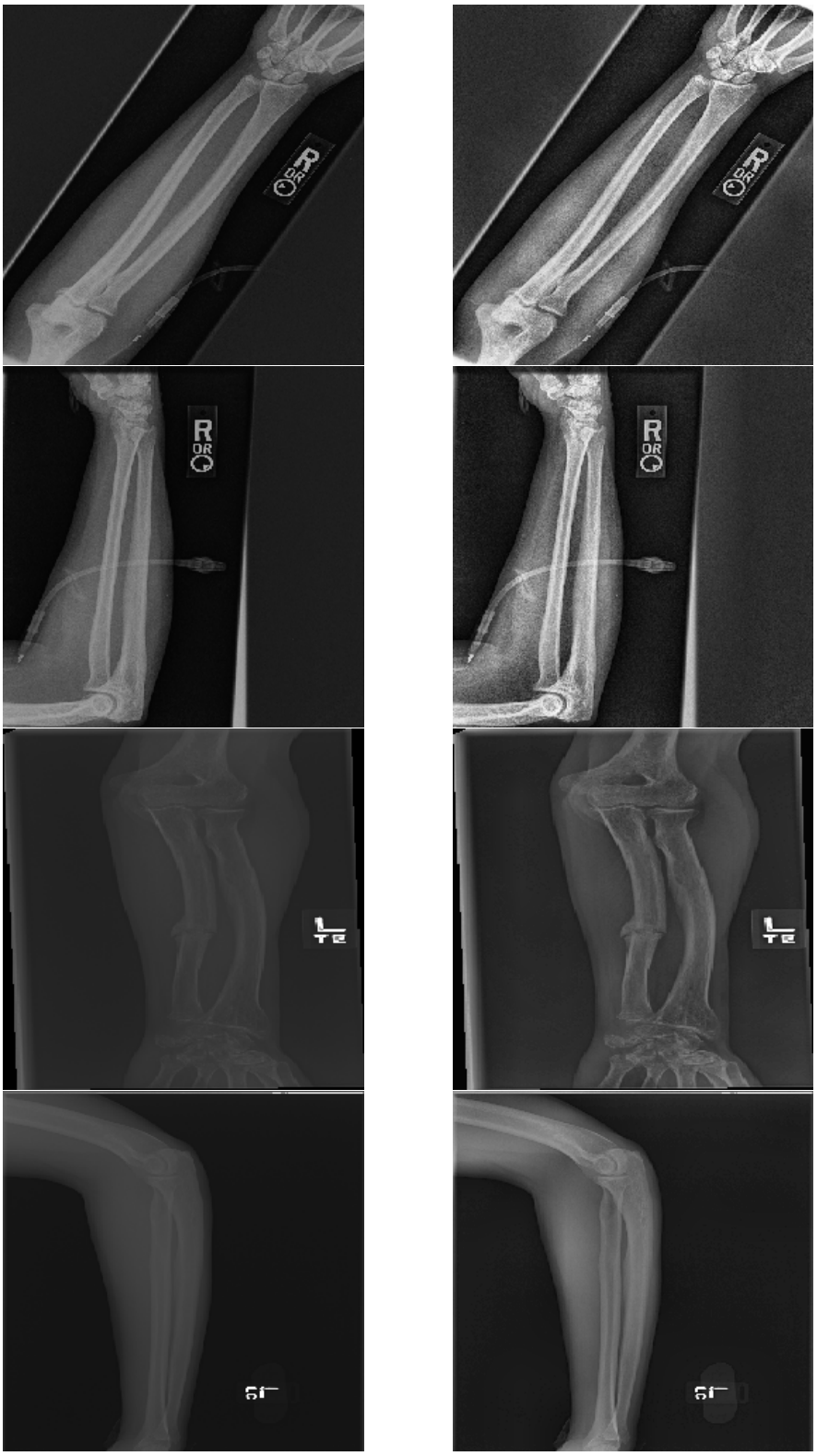


Figure 31: Images to the left are the original images and images to the right are their corresponding CLAHE activated images.

functions of the generator G_{AB} and G_{BA} can be described as follows:

$$\mathcal{L}_{GAN}(G_{AB}) = \frac{1}{2}[\sum_{b\in B}[D_A(G_{AB}(b)) - 1]^2], \tag{36}$$

$$\mathcal{L}_{GAN}(G_{BA}) = \frac{1}{2}[\sum_{a\in A}[D_B(G_{BA}(a)) - 1]^2]. \tag{37}$$

The adversarial loss functions of the discriminator D_A and D_B can be described as follows:

$$\mathcal{L}_{GAN}(D_A) = \frac{1}{2}[\sum_{a\in A}[D_A(a) - 1]^2 + \sum_{b\in B}[D_A(G_{AB}(b))]^2], \tag{38}$$

$$\mathcal{L}_{GAN}(D_B) = \frac{1}{2}[\sum_{b\in B}[D_B(b) - 1]^2 + \sum_{a\in a}[D_B(G_{BA}(a))]^2]. \tag{39}$$

It can be observed that the loss function $\mathcal{L}_{GAN}(G_{AB})$ in Equation (36) will be minimized if $D_A(G_{AB}(b))$ becomes 1. However, in Equation (38), it can also be observed that the loss $\mathcal{L}_{GAN}(D_A)$ will be minimized when $D_A(G_{AB}(b))$ becomes 0 and $D_A(a)$ becomes 1. Similarly, the loss function $\mathcal{L}_{GAN}(G_{BA})$ in Equation (37) will be minimized if $D_B(G_{BA}(a))$ becomes 1 and the loss function in Equation $\mathcal{L}_{GAN}(D_B)$ will be minimized if $D_B(G_{BA}(a))$ becomes 0 and $D_B(b)$ becomes 1. Thus, even though in this model the generators along with discriminators are trying to minimize their losses, they still remain adversaries in nature.

Cycle consistency loss and identity loss of the model were calculated in the same way as described by Equations (33) and (34). Thus, the losses can be expressed as:

$$\mathcal{L}_{cyc}(G_{AB}, G_{BA}) = \sum_{a\in A} ||G_{AB}(G_{BA}(a)) - a||_1 + \sum_{b\in B} ||G_{BA}(G_{AB}(b)) - b||_1, \tag{40}$$

$$\mathcal{L}_{identity}(G_{AB}, G_{BA}) = \sum_{a\in A} ||G_{AB}(a) - a||_1 + \sum_{b\in B} ||G_{BA}(b) - b||_1. \tag{41}$$

By combining losses described in Equations (36), (37), (40), and (41), the total loss of the of the generators can be expressed as:

$$\mathcal{L}_{G^*} = \mathcal{L}_{GAN}(G_{AB}) + \mathcal{L}_{GAN}(G_{BA}) + \lambda\mathcal{L}_{cyc}(G_{AB}, G_{BA}) + \gamma\mathcal{L}_{identity}(G_{AB}, G_{BA}). \tag{42}$$

Similarly, by combining the losses described in Equations (38) and (39), the total loss of the discriminators can be described as:

$$\mathcal{L}_{D^*} = \mathcal{L}_{GAN}(D_A) + \mathcal{L}_{GAN}(D_B). \tag{43}$$

Thus, the objective of the cycleGAN model becomes:

$$\text{Minimize}_\text{G}\ \text{Minimize}_\text{D}\ (\mathcal{L}_{G^*}, \mathcal{L}_{D^*}). \tag{44}$$

Two cycleGAN models were trained in supervised setting using the objective function described by Equation (44). One model was used to generate synthetic

X-ray images of normal forearms and the other was used to generate synthetic X-ray images of normal forearms. These translated images were generated by passing images from domain B as input to the generator G_{AB}, that is, translated images of $G_{AB}(b)\ \forall b \in B$. In total 20,000 synthetic X-ray images of forearms were generated using the models. 10,000 synthetic X-ray images of normal forearms and 10,000 synthetic X-ray images of abnormal forearms.

3.2.2.1 Choice of Generator and Discriminator Models

The architecture of the generator model was adopted from Zhu et al. (2017). The first three layers of this model were convolutional layers which were also used for downsampling the network. Downsampling was used so that the model can learn spatial features in the images more effectively. First convolutional layer used a kernel of size (7×7) and stride of 1. The remaining two convolutional layers each had kernel of size (3×3) and stride of 2. ReLU activation was used after each of the convolutional layer. 9 residual modules were added after the convoluional layers. The structure of the residual modules were similar to that of the one illustrated by Figure 17. Each of the residual module consisted of two convolutional layers each with kernel of size (3×3) and stride of 1. ReLU activation followed by dropout operation with parameter 0.2 was used after the first convolutional layer inside each residual module. After the residual modules, two inverted convolutional layers with kernels of size (3×3) and stride of 2 were used to upsample the images. $Tanh$ was used as activation function at the end of the generator model.

The architecture of the discriminator model was also adopted from Zhu et al. (2017). The discriminator model was in the form of a PatchGAN with a patch size of 70×70. PatchGAN is a type of discriminator which aims to classify whether small overlapping patches in the images are real or fake instead of classifying the whole image as real or fake [Zhu et al. (2017)]. This type of discriminator has been proven effective for images with high resolution because these models have fewer parameters. This model consisted of 6 convolutional layers. And the first 5 convolutional layers were followed by $leaky\ ReLU$ activations with α parameter valued at 0.2 for all the activations. For all the convolutional layers, the kernel was of size 4×4. However, the first 4 convolutional layers had stride of 2 whereas the last 2 convolutional layers had stride of 1.

For updating the parameters of the model, Adam optimizer was used with parameters β_1 and β_2 valued at 0.5 and 0.999 respectively and a learning rate of 0.0002. The learning rate was kept same for the first 10 epochs and then decayed linearly to zero over the next remaining epochs. The values of the parameters λ and γ in the Equation (42) were chosen to be 10 and 5 respectively.

3.2.3 Classification of Images

The objective of classification task required for this thesis was to label each image as normal or abnormal. Normal images are the ones where there is no bone defect and abnormal images are the ones where the bones are defective. To understand

the influence of adding synthetically generated images to the classification model, experiments were conducted in ablation manner. The ablation study was conducted in three stages. At first, only the natural X-ray images of forearms of MURA data were used to train and test the classification model. In the next stage a total of 10,000 synthetic X-ray images generated by the cycleGAN model was added with the natural images of MURA data. Of these 10,000 images, 5,000 were of normal forearms and 5,000 were of abnormal forearms and these images were added with the natural MURA data according to their labels. A second classification model was created using the new augmented data. And in the end, to observe influence of adding more synthetic data to the model, 10,000 more synthetic images of forearms were added where 5,000 belonged to each of the two classes. And a final classification model was created using this data.

The first classification model consisted of 1,164 normal forearm images and 661 abnormal forearm images for training. The second classification model consisted of 6,164 normal and 5,661 abnormal forearm images for training and the last classification model had 11,164 normal and 10,661 abnormal forearm images for training. However, for testing these three models, same data was used so that these models can have comparable results. The test set consisted of 150 normal images and 151 abnormal images. No synthetic data was added to the testing set so that the evaluation metrics can provide results only on natural data.

ResNet was chosen as the classification model. The first layer of the model was a convolutional layer with a kernel of size 5×5 and stride of 1. This layer was followed by, *ReLU* activation. The second layer is a max-pooling layer with a kernel of size 3×3 and stride of 2. After the max-pooling layer, 6 residual modules were added to the model. Each residual module consisted of 4 convolutional layers and two skip connections. All of the four convolutional layers had kernels of size 3×3. However, to downsample the images, first convolutional layer of each block had stride of 2 whereas the remaining 3 convolutional layers had stride of 1. *ReLU* activation was performed at the end of each residual module. After these 6 modules, a layer of average-pooling was added with a kernel of size 4×4 and stride 1 followed by a fully-connected layer with 2 output nodes. The output nodes of the fully-connected layer represents the probabilities of an image to belong to a particular class.

The architecture of the ResNet model used for creating the three models is shown in Figure 32. The change in the shape of the input data after it passes through a layer is also provided right beside that layer in the figure. It should be noted that for the sake of simplicity, activations were not shown in the figure and also n represents the number of output channels of one layer. In this model, n was initialized as 10. For updating the parameters of the model, Adam optimizer was used with parameters β_1 and β_2 valued at 0.9 and 0.999 respectively and a learning rate of 0.01. All three of the classification models were trained for 200 epochs.

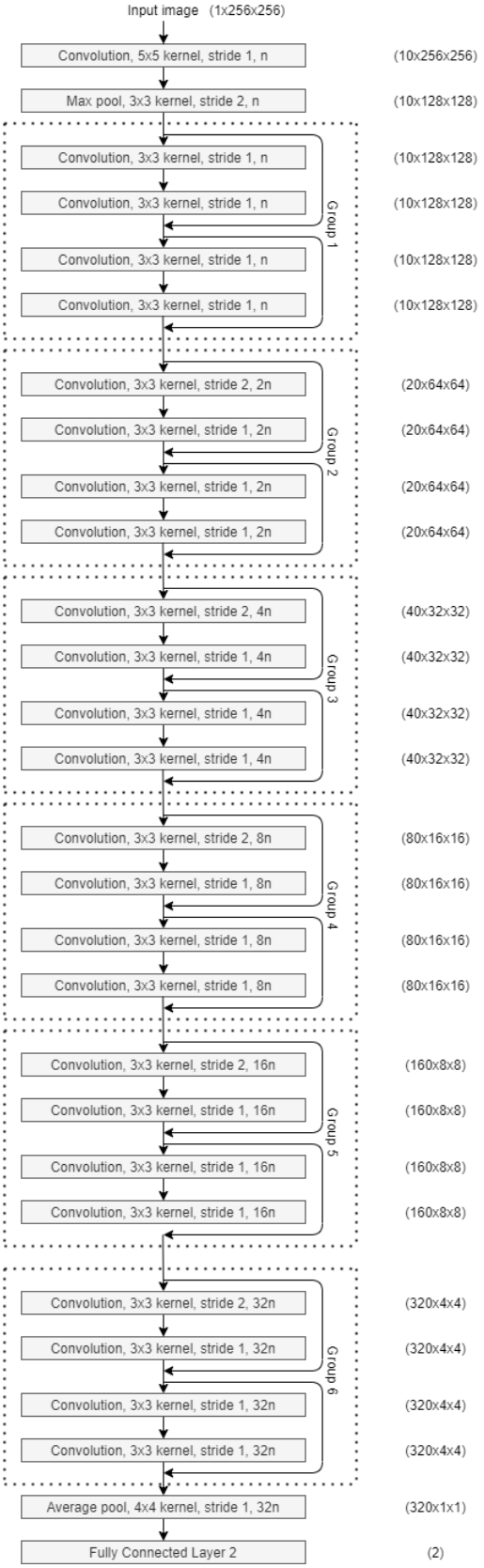


Figure 32: Architecture of the ResNet model used.

3.2.4 Image to Image Translation Using Multi-CycleGAN

The next task for this thesis was to generate segmented bones from natural X-ray images. To accomplish this task, bone segmentations of the synthetic images were extracted. It should be noted that in this thesis, in case of forearms, the resulting transformed images of cycleGAN models are referred to as synthetic images and the original Mathematica images are referred to as Mathematica images. Whereas, in case of wrists, the images created by Mathematica are referred to as synthetic images because the wrist images from Mathematica were not transformed using cycleGAN. The idea of extracting the bone segmentations from natural X-ray images by using synthetic X-ray images and their bone segmentations is similar to the idea of translating images of one domain to another. Upon successful translation of images of natural domain to images of synthetic domain, the translated images can again be translated to the domain of images with segmented bones of synthetic images. However, the cycleGAN model described in Section 2.8.4 will not be sufficient for this task since that model can accommodate image translation between two domains. To accomplish this task the basic cycleGAN model was expanded so that it can accommodate image translations among 3 domains. The expanded model was called multi-cycleGAN.

Let, A, B, and C denote the domains of natural X-ray images, synthetic X-ray images, and segmented bones of the synthetic X-ray images. For this model to translate images of each domain to the remaining two domains, 6 generators are required. Let the generators be denoted by $G_{AB}, G_{AC}, G_{BA}, G_{BC}, G_{CA}, G_{CB}$. Here the second letter of the subscript of each generator denotes the input domain and first letter of the subscript of each generator denotes the target domain, that is, G_{IJ} translates images of domain J to images of domain I. Similarly, for this model, 6 discriminators will also be required. Let the discriminators be denoted by $D_{AB}, D_{AC}, D_{BA}, D_{BC}, D_{CA}, D_{CB}$. Here, any discriminator D_{IJ} is employed to distinguish between images $\{i\}\ \forall\ i \in I$ and translated images $G_{IJ}(j)\ \forall\ j \in J$. For example, D_{AB} distinguishes between images $\{a\}\ \forall\ a \in A$ and translated images $G_{AB}(b)\ \forall\ b \in B$. The overall structure of the multi-cycleGAN model is illustrated in Figure 33.

For this model, since there are 6 generators, 6 adversarial loss functions are required for the generators which are similar to the ones described by the Equations (36) and (37). The adversarial loss functions for the generators are as follows:

$$\mathcal{L}_{GAN}(G_{AB}) = \frac{1}{2}[\sum_{b\in B}[D_{AB}(G_{AB}(b)) - 1]^2], \tag{45}$$

$$\mathcal{L}_{GAN}(G_{BA}) = \frac{1}{2}[\sum_{a\in A}[D_{BA}(G_{BA}(a)) - 1]^2], \tag{46}$$

$$\mathcal{L}_{GAN}(G_{AC}) = \frac{1}{2}[\sum_{c\in C}[D_{AC}(G_{AC}(c)) - 1]^2], \tag{47}$$

$$\mathcal{L}_{GAN}(G_{CA}) = \frac{1}{2}[\sum_{a\in A}[D_{CA}(G_{CA}(a)) - 1]^2], \tag{48}$$

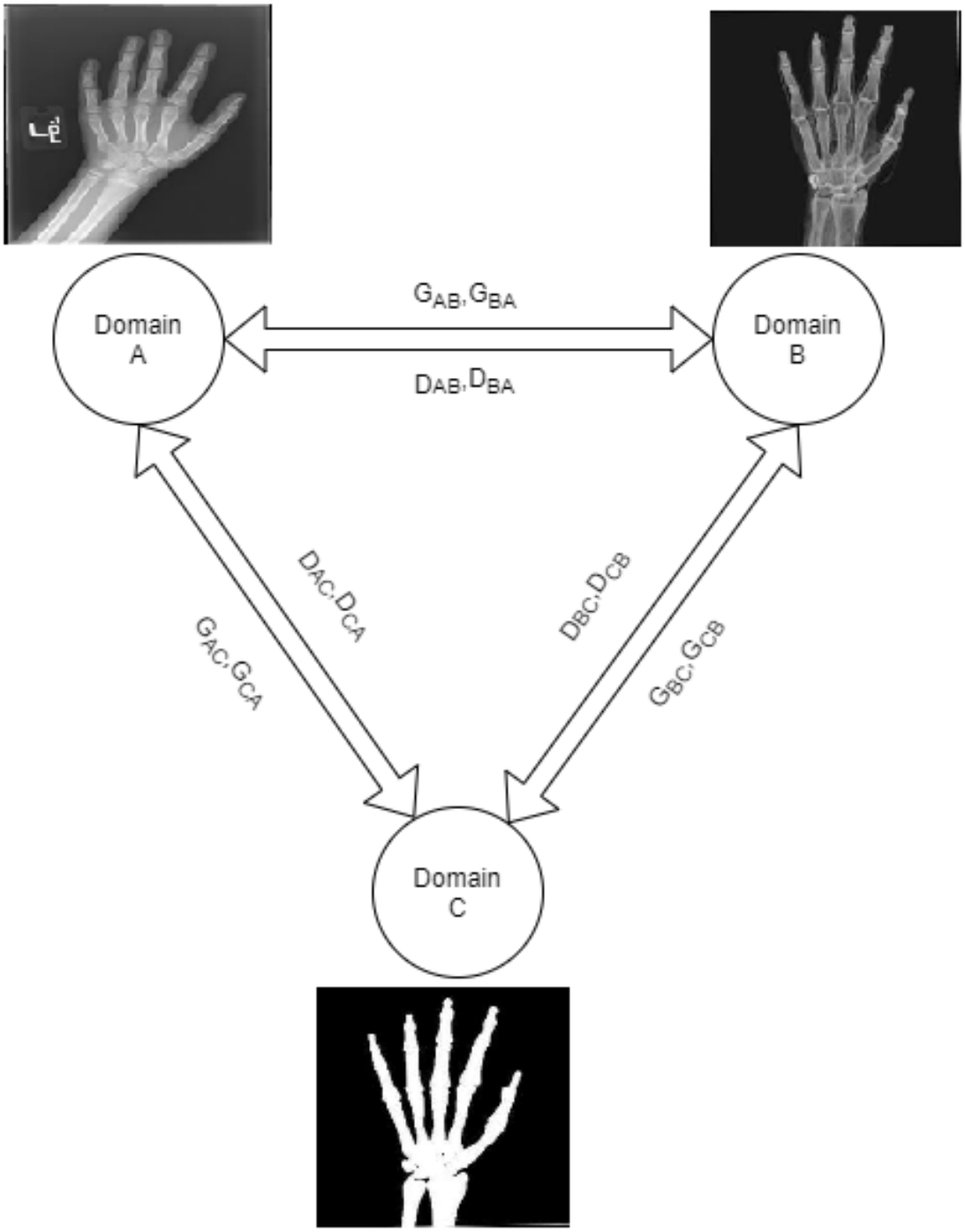


Figure 33: Structure of the multi-cycleGAN model. In the figure, each circle represents a domain and the image near a circle is an example image of that domain. Each pair of domains is associated with two generators and two discriminators.

$$\mathcal{L}_{GAN}(G_{BC}) = \frac{1}{2}[\sum_{c \in C}[D_{BC}(G_{BC}(c)) - 1]^2], \tag{49}$$

$$\mathcal{L}_{GAN}(G_{CB}) = \frac{1}{2}[\sum_{b\in B}[D_{CB}(G_{CB}(b)) - 1]^2]. \tag{50}$$

Similarly, for the 6 discriminators of the model, 6 adversarial loss functions are required. These loss functions are similar to the ones described in (38) and (39) and can be described as follows:

$$\mathcal{L}_{GAN}(D_{AB}) = \frac{1}{2}[\sum_{a\in A}[D_{AB}(a) - 1]^2 + \sum_{b\in B}[D_{AB}(G_{AB}(b))]^2], \tag{51}$$

$$\mathcal{L}_{GAN}(D_{BA}) = \frac{1}{2}[\sum_{b\in B}[D_{BA}(b) - 1]^2 + \sum_{a\in A}[D_{BA}(G_{BA}(a))]^2], \tag{52}$$

$$\mathcal{L}_{GAN}(D_{AC}) = \frac{1}{2}[\sum_{a\in A}[D_{AC}(a) - 1]^2 + \sum_{c\in C}[D_{AC}(G_{AC}(c))]^2], \tag{53}$$

$$\mathcal{L}_{GAN}(D_{CA}) = \frac{1}{2}[\sum_{c\in C}[D_{CA}(c) - 1]^2 + \sum_{a\in A}[D_{CA}(G_{CA}(a))]^2], \tag{54}$$

$$\mathcal{L}_{GAN}(D_{BC}) = \frac{1}{2}[\sum_{b\in V}[D_{BC}(b) - 1]^2 + \sum_{c\in C}[D_{BC}(G_{BC}(c))]^2], \tag{55}$$

$$\mathcal{L}_{GAN}(D_{CB}) = \frac{1}{2}[\sum_{c\in C}[D_{CB}(c) - 1]^2 + \sum_{b\in B}[D_{CB}(G_{CB}(b))]^2]. \tag{56}$$

This model will also need three cycle consistency loss functions similar to the one described by Equation (40). The cycle consistency loss functions are as follows:

$$\mathcal{L}_{cyc}(G_{AB}, G_{BA}) = \sum_{a\in A} ||G_{AB}(G_{BA}(a)) - a||_1 + \sum_{b\in B} ||G_{BA}(G_{AB}(b)) - b||_1, \tag{57}$$

$$\mathcal{L}_{cyc}(G_{CA}, G_{AC}) = \sum_{c\in C} ||G_{CA}(G_{AC}(c)) - c||_1 + \sum_{a\in A} ||G_{AC}(G_{CA}(a)) - a||_1, \tag{58}$$

$$\mathcal{L}_{cyc}(G_{BC}, G_{CB}) = \sum_{b\in B} ||G_{BC}(G_{CB}(b)) - b||_1 + \sum_{c\in C} ||G_{CB}(G_{BC}(c)) - c||_1. \tag{59}$$

And finally, this model will need three identity loss functions similar to the one described by Equation (41) which are described as follows:

$$\mathcal{L}_{identity}(G_{AB}, G_{AC}) = \sum_{a\in A}[||G_{AB}(a) - a||_1 + ||G_{AC}(a) - a||_1], \tag{60}$$

$$\mathcal{L}_{identity}(G_{BA}, G_{BC}) = \sum_{b\in B}[||G_{BA}(b) - b||_1 + ||G_{BC}(b) - b||_1], \tag{61}$$

$$\mathcal{L}_{identity}(G_{CB}, G_{CA}) = \sum_{c\in C}[||G_{CB}(c) - c||_1 + ||G_{CA}(c) - c||_1]. \tag{62}$$

Thus, by combining all the 12 losses of generators, the total generator loss $\mathcal{L}(G^*)$ for this model can be expressed as:

$$\begin{aligned}\mathcal{L}(G^*) = \mathcal{L}_{GAN}(G_{AB}) + \mathcal{L}_{GAN}(G_{BA}) + \mathcal{L}_{GAN}(G_{AC}) + \mathcal{L}_{GAN}(G_{CA}) \\ + \mathcal{L}_{GAN}(G_{BC}) + \mathcal{L}_{GAN}(G_{CB}) \\ + \lambda[\mathcal{L}_{cyc}(G_{AB}, G_{BA}) + \mathcal{L}_{cyc}(G_{AC}, G_{CA}) + \mathcal{L}_{cyc}(G_{BC}, G_{CB})] \\ + \gamma[\mathcal{L}_{identity}(G_{AB}, G_{AC}) + \mathcal{L}_{identity}(G_{BA}, G_{BC}) + \mathcal{L}_{identity}(G_{CB}, G_{CA})].\end{aligned} \tag{63}$$

Similarly, by combining all the 6 losses of discriminators, the total discriminator loss $\mathcal{L}(D^*)$ for this model can be expressed as:

$$\begin{aligned}\mathcal{L}(D^*) = \mathcal{L}_{GAN}(D_{AB}) + \mathcal{L}_{GAN}(D_{BA}) + \mathcal{L}_{GAN}(D_{AC}) + \mathcal{L}_{GAN}(D_{CA}) \\ + \mathcal{L}_{GAN}(D_{BC}) + \mathcal{L}_{GAN}(D_{CB}).\end{aligned} \tag{64}$$

Thus, the objective of the multi-cycleGAN model becomes:

$$\text{Minimize}_{\text{G}}\ \text{Minimize}_{\text{D}}\ (\mathcal{L}_{G^*}, \mathcal{L}_{D^*}). \tag{65}$$

The choice of generators and discriminators for the multi-cycleGAN model was same as the ones described in Section 3.2.2.1. However, during experimentation it was observed that the discriminators were learning much faster than the generators so, to circumvent the issue and stabilize the model, two techniques were employed. Firstly, for every updating of gradients of discriminators, the gradients of generators were updated 3 times. Secondly, separate optimizers were used for generators and discriminators. Adam optimizer with parameters β_1 and β_2 valued at 0.5 and 0.9 respectively and a learning rate of 0.0002 were used for the generators whereas for the discriminators, while the values of β_1 and β_2 were kept the same, the learning rate of the Adam optimizer was reduced to 0.00005. Both of these learning rates were kept same for the first 10 epochs and then decayed linearly to zero over the next remaining epochs. The values of the parameters λ and γ in the Equation (63) were chosen to be 5 and 1 respectively.

After training, to segment bones from natural X-ray images, two translation methods were used. Let y denote the resulting segmented images. Then the two methods can be expressed as:

$$y = G_{CB}(G_{BA}(a)) \qquad \forall\ a \in A, \tag{66}$$

$$y = G_{CB}(a) \quad \forall\ a \in A. \tag{67}$$

In the method described by Equation (66), the input image a goes through two successive translations. In the first translation, the image gets translated to domain B which is the domain of synthetic images then it gets translated to domain C which is the domain of bone segmented images. Thus, after the second translation, segmented images of bones are generated. In the method described by Equation (67), the input image a goes through one translation. Here, the generator G_{CB} treats the image a as a sample from domain B and translates it to domain C. Thus, generating images of segmented bones.

4 Results

In this section, detailed results of all the experiments described in Section 3.2 are provided and discussed.

4.1 CycleGAN Images

Two cycleGAN models were created to transform images of synthetic domain to natural domain. One model was trained with normal forearm images extracted from Mathematica as inputs to synthetic domain and normal forearm images of MURA dataset as inputs to natural domain. The other model was trained with abnormal forearm images extracted from Mathematica as source of synthetic domain and abnormal forearm images of MURA dataset as source of natural domain. Both these models were trained in supervised setting as we required transformed images to belong to a specific class so that we can add those images to the classification model with proper labels. The objective function for both of the model is given in Equation (44). The architecture of the generator and discriminator models along with the choice of their parameters are described in Section 3.2.2.1.

Some examples of results obtained by the model that transformed normal Mathematica forearm images to the domain of normal forearm images of MURA data is given in Figure 34. Similar examples of results obtained by the model which transformed abnormal Mathematica forearm images to the domain of abnormal MURA forearm images are shown in Figure 35. Both of these models were trained for 30 epochs to generate the synthetic images. It was observed that if the models were trained for many more epochs, the quality of the generated images started to deteriorate. It can be observed that the cycleGAN models tried to generate soft tissues around the skeleton of the synthetic images to mimic real X-ray images. The model tried to add white borders in the images that are similar to the natural X-ray images. And on the top left side of the soft tissues, the model has created some bright spots trying to mimic the annotation letters such as L, R in the original X-ray images. It can also be observed in Figure 35 that the cycleGAN model kept the skeleton abnormalities intact.

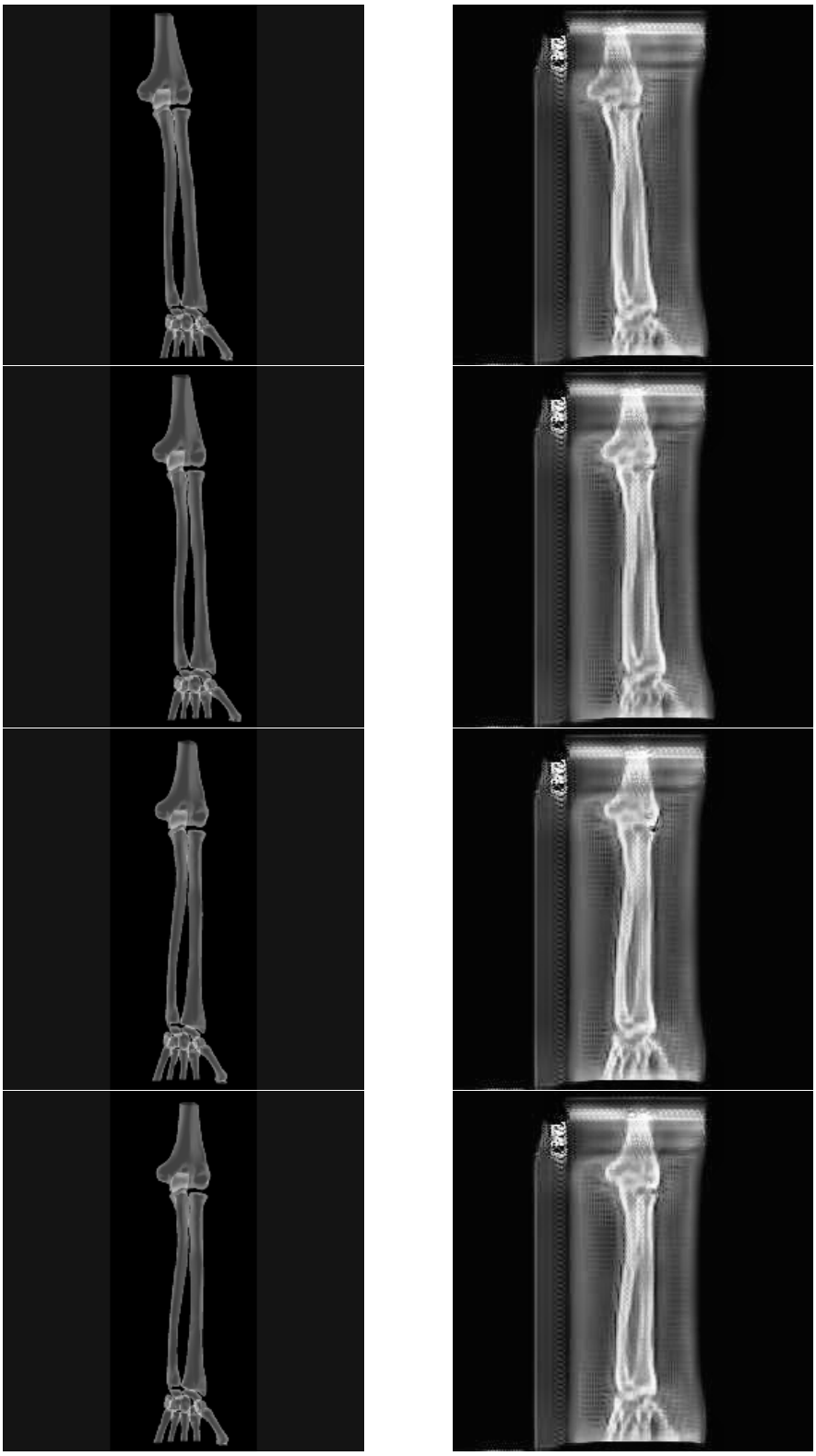


Figure 34: Images to the left are normal Mathematica images and images to the right are their corresponding transformed normal synthetic images.

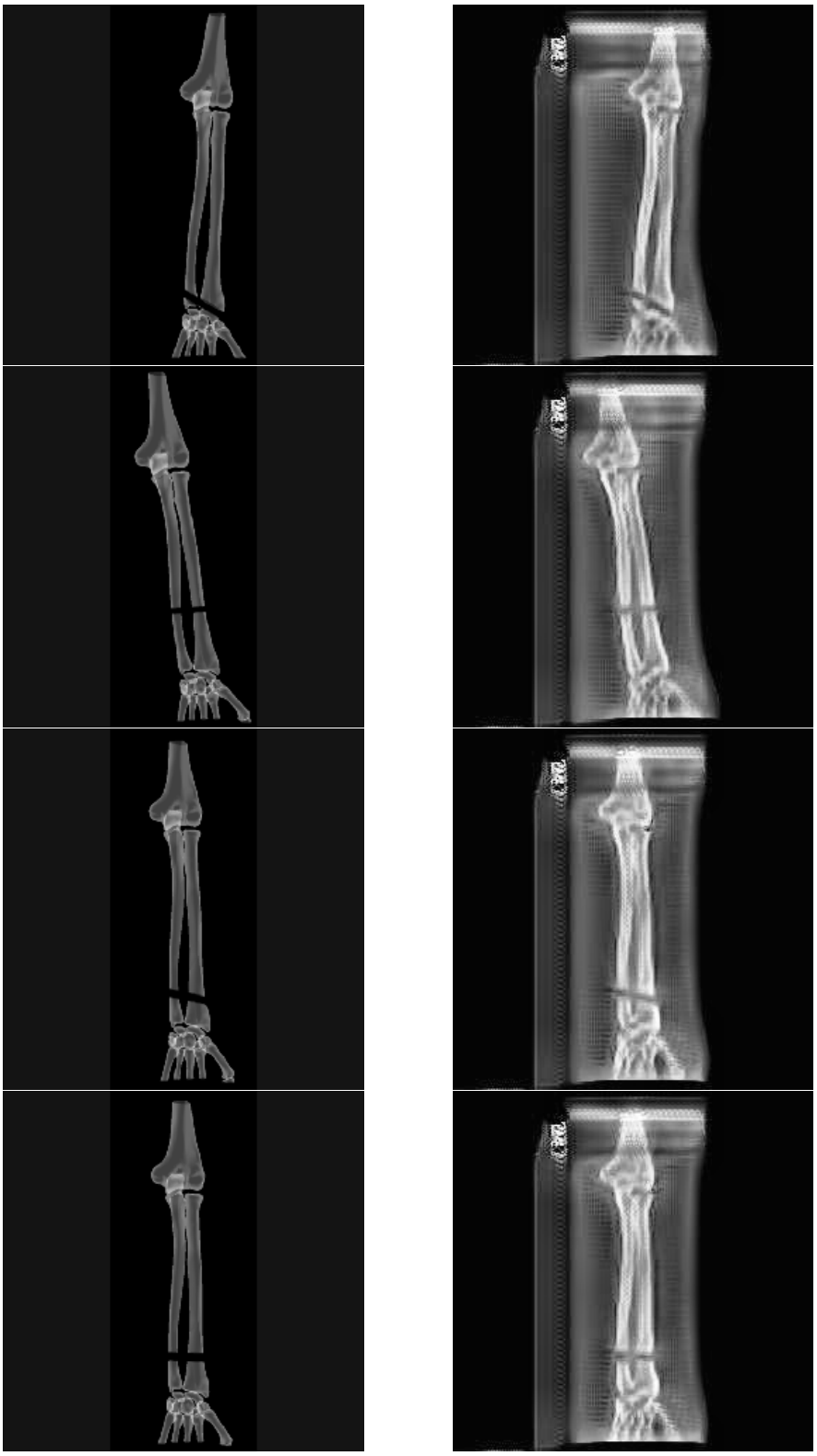


Figure 35: Images to the left are abnormal Mathematica images and images to the right are their corresponding transformed abnormal synthetic images.

4.2 Classification of Normal and Abnormal Bones

Sensitivity, specificity, and accuracy were chosen as the evaluation metrics to evaluate the performance of the three different classification models described in Section 3.2.3. Table 2 summarises the best observed results.

Dataset	Sensitivity	Specificty	Accuracy
Natural data only	0.623	0.693	0.658
Natural data + 5,000 synthetic images per class	0.629	0.847	0.787
Natural data + 10,000 synthetic images per class	0.675	0.86	0.767

Table 2: Sensitivity, specificity, and accuracy scores of all the classifiers evaluated on the test dataset.

Upon repeating the training of the 3 different models described in Table 2 for 5 times each, it was also observed that the standard deviations of accuracy for the first, second, and third models were 0.03, 0.004, and 0.003 respectively. Which implies that including the synthetic data helps the model stabilize significantly. The results in Table 2 also show that in best cases, sensitivity was increased by 0.05, specificity was increased by 0.15, and accuracy was increased by 0.13. This change in performance is significant, which suggests that by adding synthetic data the performance of the models can be improved significantly. It can also be observed that by adding synthetic data, specificity of the model improved significantly more than sensitivity of the model. This suggests that the model was able to classify images of negative class more accurately than images of positive class which means X-ray images with no bone abnormalities were more accurately classified than images with bone abnormalities. This was expected because the abnormalities introduced to the synthetic images were very different than abnormalities in the natural X-ray images.

In MURA dataset abnormalities included fractured bones, dislocated bones, metallic pins attached to the bones whereas in the synthetic data the only abnormality was fractured bones. This might have led to better classification of images with fractures which in turn increased the sensitivity of the model slightly. From the results in Table 2 it can also be concluded that the third classification model which had 10,000 additional synthetic images in each class did not increase the overall performance of the second model which had 5,000 additional synthetic images in each class. The decrement of accuracy by a value of 0.02 from second to third model suggests that adding more synthetic data will not improve the classification model but might decrease its performance.

The training and testing accuracy progress graph for the model with only natural X-ray images are shown in Figure 36. It can be observed from the graph that the model did not stabilize after 200 epochs. The training and testing accuracy progress graph for the model where in each class along with the natural X-ray images 5,000 synthetic images were added are shown in Figure 37. And Figure 38 visualizes a similar graph for the model where in each class along with the natural X-ray images

10,000 synthetic images were added. In both Figures 37 and 38 it can be observed that the model has stabilized and converged on the test data after around 100 epochs. It can also be observed that training accuracy is significantly higher than testing accuracy. This is not a result of overfitting but the result of synthetic images being very similar to each other. And due to this similarity, during training, the model classifies the synthetic images accurately and since the data for testing the model does not have any synthetic image, the performance drops.

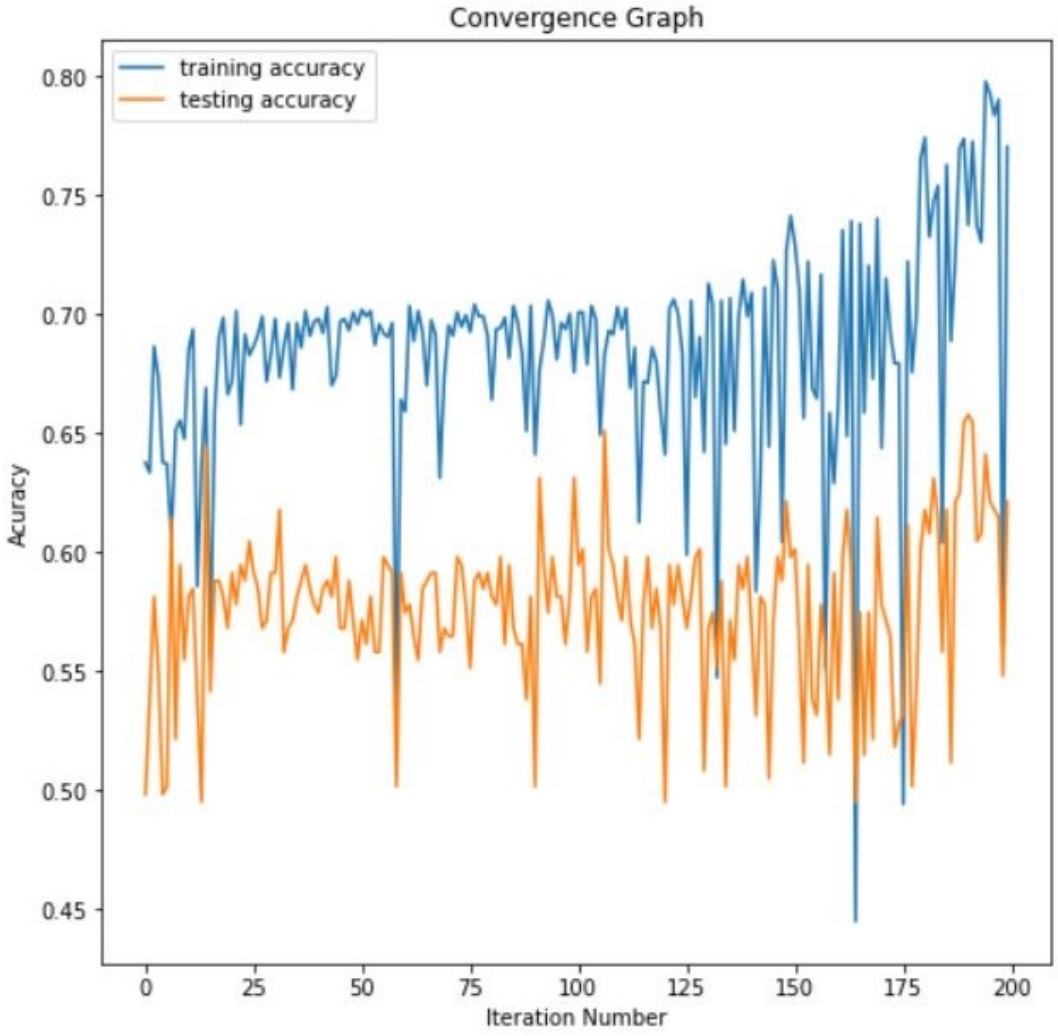


Figure 36: Convergence graph of the model with only natural X-ray images.

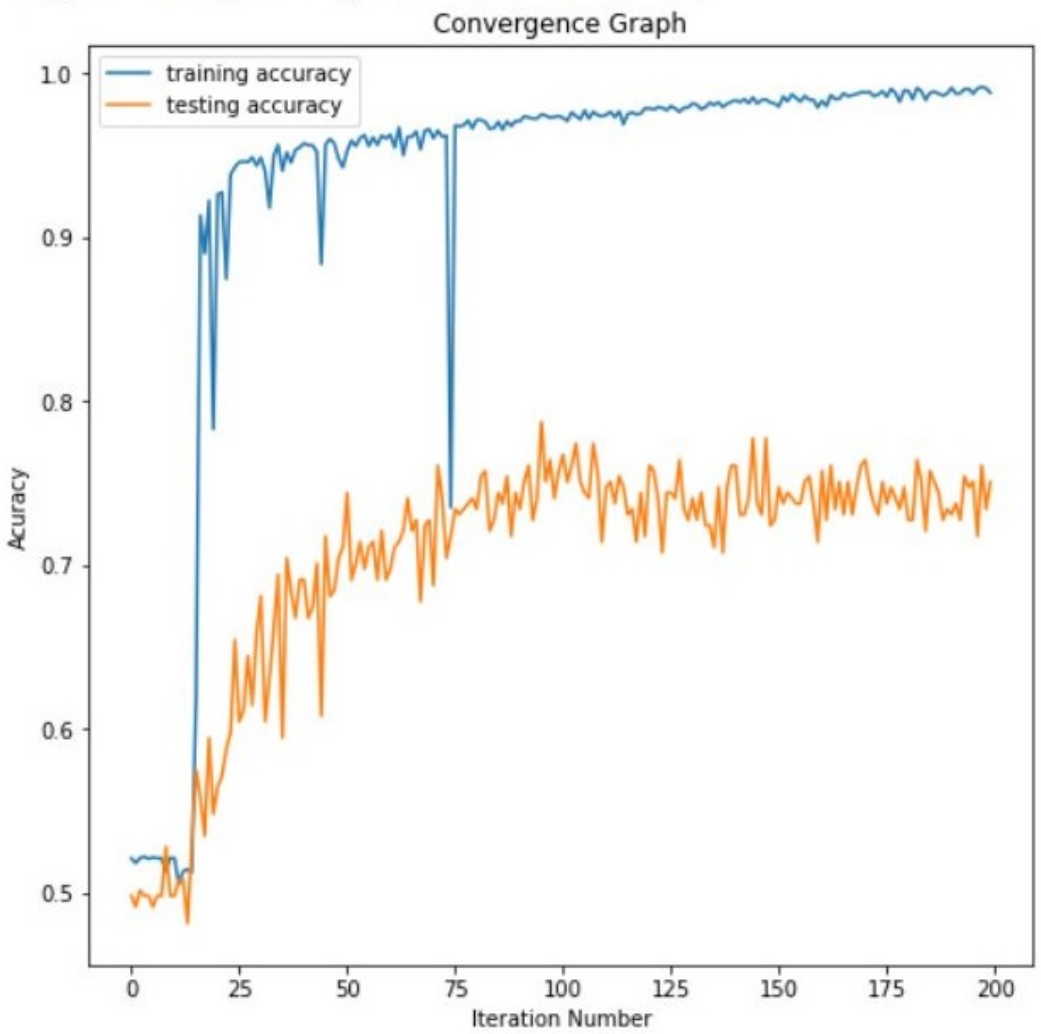


Figure 37: Convergence graph of the model with natural X-ray images and 5,000 synthetic images in each class.

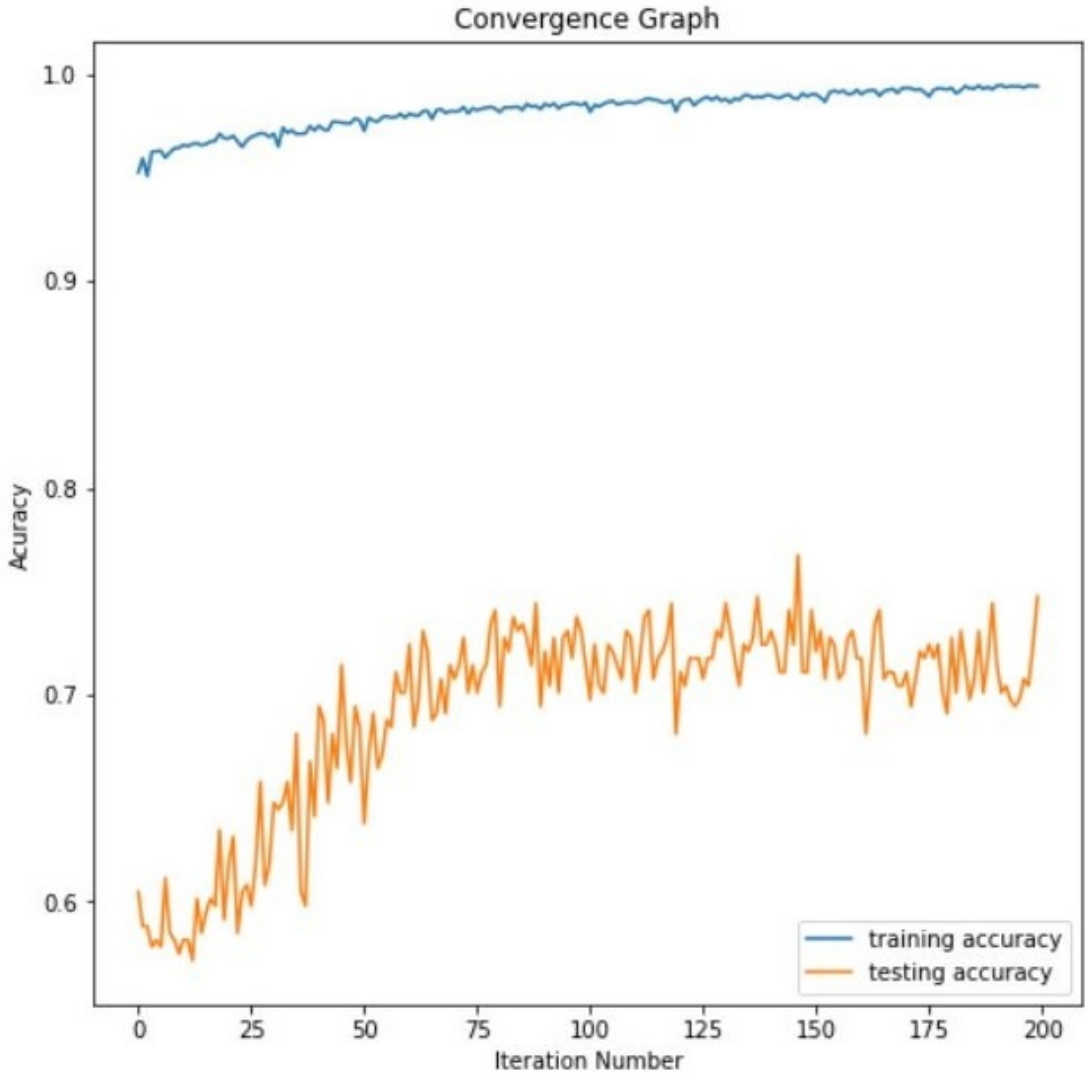


Figure 38: Convergence graph of the model with natural X-ray images and 10,000 synthetic images in each class.

4.3 Generating Bone Segmentations

Two multi-cycleGAN models were trained to extract bones from X-ray images. The architecture of generator and discriminator models along with the parameters on these multi-cycleGANs are described in Section 3.2.4. The first model consisted of X-ray images of forearms from MURA data as input to domain A, synthetic images of forearms as input to domain B, and bone segmentations of synthetic images of forearms as input to domain C. The second model was trained using X-ray images of wrists from RSNA data as input to domain A, synthetic images of wrists as input to domain B, and bone segmentations of synthetic images of wrists as input to domain C. Both the methods described by Equation (66) and Equation (67) were used to generate images from both of these models. It was observed that method described by Equation (67) produced better results for MURA data and method described by Equation (66) performed better for RSNA data. This is because the images in MURA data is very different from the synthetic forearm images. Thus, when method described by Equation (66) was used on the MURA data, the model tried to translate these original images to the domain of synthetic forearm images then tried to extract their segmented bones and this is a task too difficult for the model. Instead, when the method described by Equation (67) was used for MURA data, the generator GCB treated these images as images of domain B and translated them to domain C. Thus, generating the images of segmented bones.

In case of RSNA data, the original X-ray images were very similar to the synthetic X-ray images. Thus, the task of translating these original images to the domain of synthetic images was not a very difficult task and then these translated images could also be translated to the domain of segmented images. This is why for the RSNA data the method described by (66) performed well.

4.3.1 Bone Segmentations of MURA Data

Examples of annotations generated by using natural forearm X-ray images of MURA, synthetic forearm X-ray images, and bone segmentation of synthetic forearm images to train multi-cycleGAN model are given in Figures 39 and 40. The images were generated by training the model for 30 epochs and by using the method described by Equation (67) to generate images. It was also observed that upon training the model for too many epochs, the quality of the segmented images started to degrade.

It can be observed that the model was not able to segment the bones properly. However, it did segment majority of the portions of the bones correctly and upon close observation it can be seen that the model was also able to segment some parts of the metallic pins are are attached to some of the bones. The model completely removed the soft tissues around the bones and it was also able to remove parts of letter annotations from the images as well. However, in some cases the model did produce segmented boundaries of the X-ray plates as it can been seen in the images.

4.3.2 Bone Segmentations of RSNA Data

Examples of annotations generated by using natural wrist X-ray images of RSNA, synthetic wrist X-ray images of Mathematica, and bone segmentation of synthetic wrist images to train the multi-cycleGAN is given in Figure 41. Similar to the previous multi-cycleGAN model, this one was also trained for 30 epochs. However, in this case the images were generated using the Equation (66). Also, similar to the previous multi-cycleGAN model, it was observed that if this model is trained for too many epochs, the quality of the segmented images degrade.

It can be observed that the model performed very well to segment bones from the X-ray images when it is compared to the models were MURA data was used. The model was able to completely remove soft tissues around the bones. The model also removed the letter annotations from the images. However, even though the model was able to capture the overall positioning of the wrist and was able to generate annotations accordingly, it was not able to segment the bones on a local level. It can be seen that the segmented bones do not represent the position of the fingers in accordance with the input images. In the segmented images the angles between the fingers as well the size of the fingers are different, when compared to the original images. However, it was also observed that the model was able to segment the bones of the forearm correctly from the images.

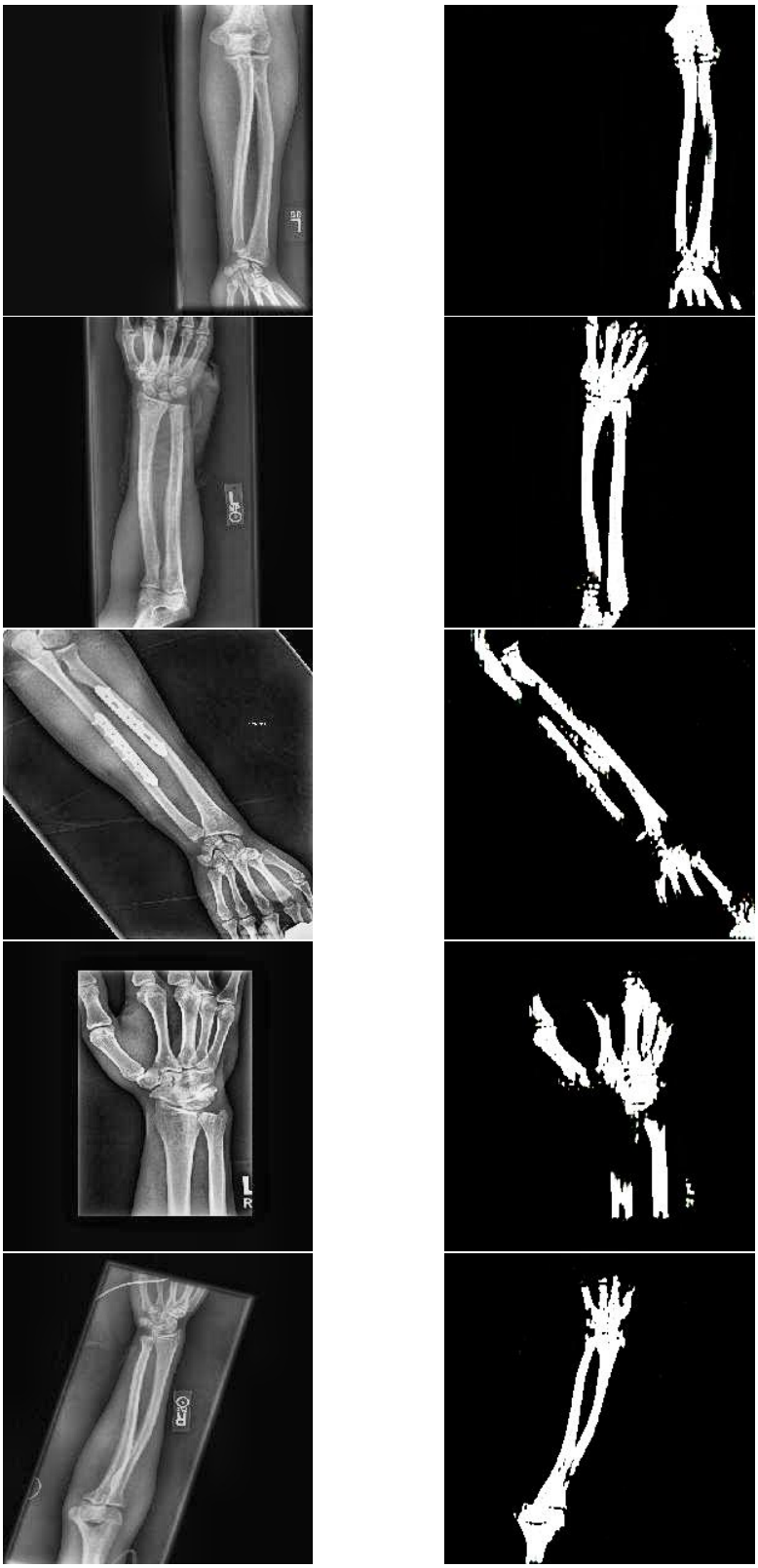


Figure 39: Images to the left are normal forearm images of MURA and images to the right are their corresponding segmented images.

Figure 40: Images to the left are abnormal forearm images of MURA and images to the right are their corresponding segmented images.

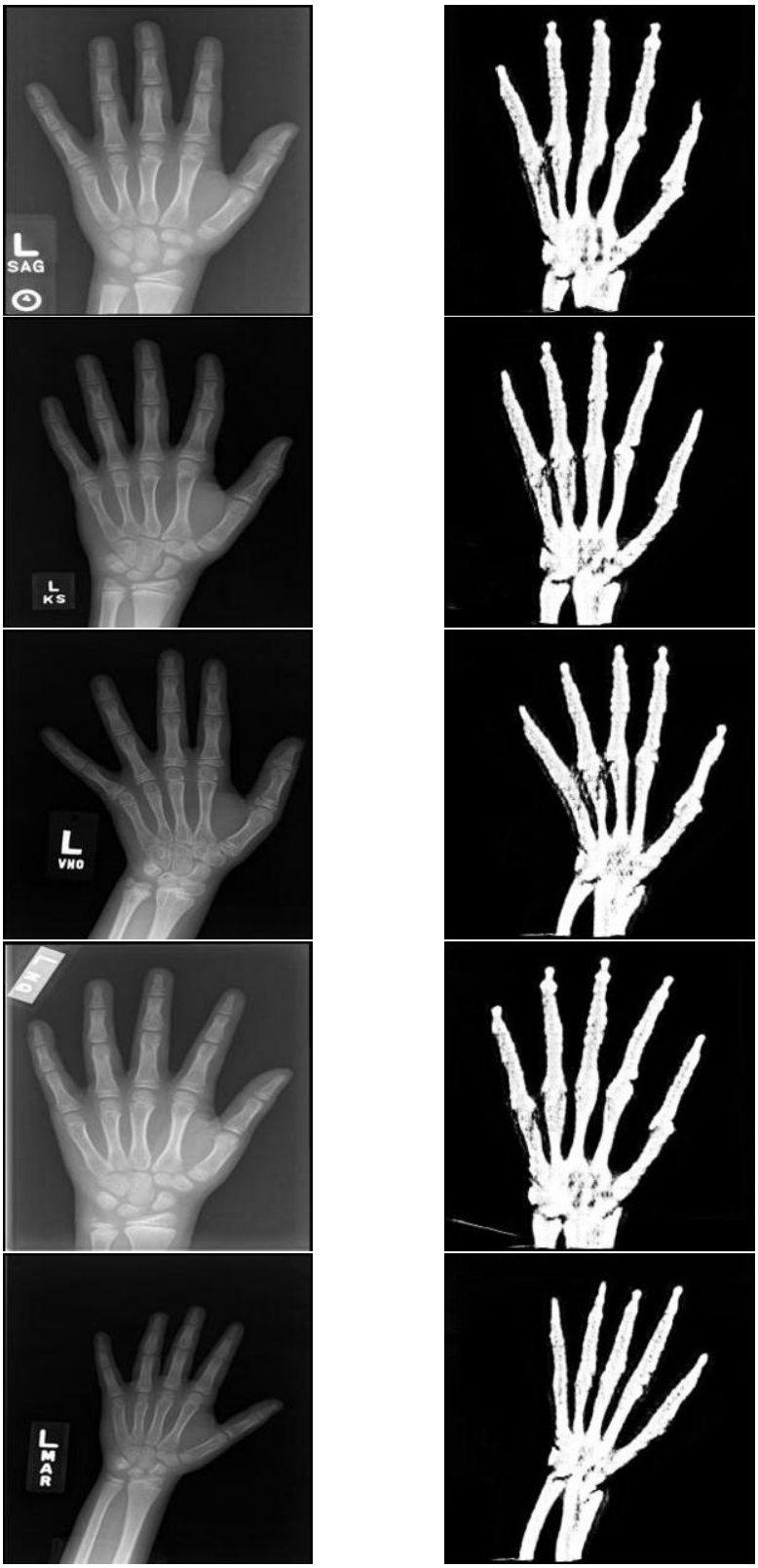


Figure 41: Images to the left are wrist images of RSNA and images to the right are their corresponding segmented images.

5 Conclusion

5.1 Summary

This thesis studied ways to reduce amount of natural data needed to generate annotations from X-ray images and also to classify defects in bones. To circumvent the issues of unavailability of sufficient data for classification of bone defects, cycleGAN models were trained to generate synthetic data. The cycleGAN models were trained in supervised manner so that each model can generate synthetic images from a particular class. The generated synthetic images were used to augment the MURA dataset which does not have sufficient amount of images to train a deep neural model appropriately. Classification models were trained and tested on MURA dataset before and after data augmentation.

The cycleGAN model was expanded so that it may accommodate image translation among images of 3 domains. This model was called multi-cycleGAN. A method was proposed using the multi-cycleGAN models to generate images of segmented bones from natural X-ray images. One multi-cycleGAN model was trained on MURA data where it was accompanied by synthetic X-ray images of forearms and bone segmentations of the X-ray images. This model was used to generate segmented bones of X-ray images of forearms in MURA data. Another multi-cycleGAN model was trained using RSNA X-ray images of wrists, synthetic X-ray images of wrists and bone segmentations of the synthetic X-ray images. This model was used to extract segmented bones of wrists from X-ray images in RSNA data.

Significant improvement of classification results were observed after synthetic data was augmented with MURA data. Results of bone segmentations from X-ray images of forearms in MURA data were not good enough to be used for medical diagnosis. Results of bone segmentations from X-ray images of wrists in RSNA data were much better. It was observed that images of segmented bones from wrists were consistent with natural X-ray images in overall shape of the wrist but inconsistent in angles between fingers and size of fingers.

5.2 Discussion and Future Work

From the results it can be concluded that MURA dataset is not a good dataset for generating annotations as the quality of the images is very poor. When RSNA data was used to generate annotations, much better results were observed. However, typical generative models that produce good results consist of much more data than what was available for this thesis. With availability of more data, better models with larger architectures can be created. It can also be concluded that using cycleGAN for data augmentation is an effective method to improve performance of classification models.

There remains many aspects which could be investigated in future works. A more extensive search can be conducted to find more appropriate parameters for the GAN models as well as the classification models. The generated segmented images by using multi-cycleGAN models may also be used to calculate features like

radial inclination or volar tilt. More variants of model architectures of generator and discriminator models of cycleGAN and multi-cycleGAN can be experimented with for the purpose of generating better annotations. Also, in future novel GAN models can be implemented that can translate more specific features from one domain of images to another. These GAN models then can be used to propagate more specific annotations in synthetic images to natural images appropriately.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Aggarwal, C. C. (2018). Neural Networks and Deep Learning. *Springer*, 10:978–3.

Akkus, Z., Galimzianova, A., Hoogi, A., Rubin, D. L., and Erickson, B. J. (2017). Deep learning for brain MRI segmentation: state of the art and future directions. *Journal of Digital Imaging*, 30(4):449–459.

Alpaydin, E. (2020). *Introduction to Machine Learning*. MIT press.

Avendi, M., Kheradvar, A., and Jafarkhani, H. (2016). A combined deep-learning and deformable-model approach to fully automatic segmentation of the left ventricle in cardiac MRI. *Medical Image Analysis*, 30:108–119.

Beutel, J., Kundel, H. L., and Van Metter, R. L. (2000). *Handbook of Medical Imaging*, volume 1. Spie Press.

Bishop, C. M. (1995). *Neural Networks for Pattern Recognition*. Oxford University Press.

Bisong, E. (2019). Google colaboratory. In *Building Machine Learning and Deep Learning Models on Google Cloud Platform*, pages 59–64. Springer.

Carpenter, K. A., Cohen, D. S., Jarrell, J. T., and Huang, X. (2018). Deep learning and virtual drug screening. *Future Medicinal Chemistry*, 10(21):2557–2567.

Chai, H. Y., Wee, L. K., Swee, T. T., and Hussain, S. (2011). Gray-level co-occurrence matrix bone fracture detection. *WSEAS Transactions on Systems*, 10(1):7–16.

Chollet, F. et al. (2015). Keras. https://keras.io. [Accessed: 22-April-2021].

Cireşan, D. C., Giusti, A., Gambardella, L. M., and Schmidhuber, J. (2013). Mitosis detection in breast cancer histology images with deep neural networks. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 411–418. Springer.

Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., and Darrell, T. (2014). Decaf: A deep convolutional activation feature for generic visual recognition. In *International Conference on Machine Learning*, pages 647–655. PMLR.

Dougherty, G. (2011). *Medical Image Processing: Techniques and Applications*. Springer Science & Business Media.

Fuhl, W., Geisler, D., Rosenstiel, W., and Kasneci, E. (2019). The applicability of cycle GANs for pupil and eyelid segmentation, data generation and image refinement. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 4406–4415.

Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 580–587.

Gonzalez, R. C. and Woods, R. E. (2008). *Digital Image Processing*. Prentice Hall, Upper Saddle River, N.J.

Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y. (2016). *Deep Learning*. MIT press Cambridge.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc.

Gulshan, V., Peng, L., Coram, M., Stumpe, M. C., Wu, D., Narayanaswamy, A., Venugopalan, S., Widner, K., Madams, T., Cuadros, J., et al. (2016). Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs. *Journal of the American Medical Association*, 316(22):2402–2410.

Halabi, S. S., Prevedello, L. M., Kalpathy-Cramer, J., Mamonov, A. B., Bilbily, A., Cicero, M., Pan, I., Pereira, L. A., Sousa, R. T., Abdala, N., et al. (2019). The RSNA pediatric bone age machine learning challenge. *Radiology*, 290(2):498–503.

Harisinghani, M. G., Chen, J. W., and Weissleder, R. (2018). *Primer of Diagnostic Imaging*. Elsevier Health Sciences.

Haykin, S. S. (2009). Neural Networks and Learning Machines.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778.

Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366.

Huo, Y., Xu, Z., Moon, H., Bao, S., Assad, A., Moyo, T. K., Savona, M. R., Abramson, R. G., and Landman, B. A. (2018). SynSeg-net: Synthetic segmentation without target modality ground truth. *IEEE Transactions on Medical Imaging*, 38(4):1016–1025.

Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Jha, A. K., Larizgoitia, I., Audera-Lopez, C., Prasopa-Plaizier, N., Waters, H., and Bates, D. W. (2013). The global burden of unsafe medical care: analytic modelling of observational studies. *BMJ Quality & Safety*, 22(10):809–815.

Kingma, D. and Ba, J. (2014). Adam: A method for stochastic optimization. *International Conference on Learning Representations*.

LeCun, Y. A., Bottou, L., Orr, G. B., and Müller, K.-R. (2012). Efficient backprop. In *Neural Networks: Tricks of the trade*, pages 9–48. Springer.

Li, M., Zhang, T., Chen, Y., and Smola, A. J. (2014). Efficient mini-batch training for stochastic optimization. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 661–670.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. (2016). SSD: Single shot multibox detector. In *European Conference on Computer Vision*, pages 21–37. Springer.

Liu, W., Cheng, L., and Meng, D. (2018). Brain slices microscopic detection using simplified SSD with cycle-GAN data augmentation. In *International Conference on Neural Information Processing*, pages 454–463. Springer.

Mao, X., Li, Q., Xie, H., Lau, R. Y., Wang, Z., and Paul Smolley, S. (2017). Least squares generative adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2794–2802.

Meyer-Bäse, A., Meyer-Baese, A., and Schmid, V. J. (2004). *Pattern Recognition and Signal Analysis in Medical Imaging*. Academic Press.

Mirza, M. and Osindero, S. (2014). Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Perez, L. and Wang, J. (2017). The effectiveness of data augmentation in image classification using deep learning. *arXiv preprint arXiv:1712.04621*.

Pizer, S. M., Amburn, E. P., Austin, J. D., Cromartie, R., Geselowitz, A., Greer, T., ter Haar Romeny, B., Zimmerman, J. B., and Zuiderveld, K. (1987). Adaptive histogram equalization and its variations. *Computer Vision, Graphics, and Image Processing*, 39(3):355–368.

Radford, A., Metz, L., and Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. In Bengio, Y. and LeCun, Y., editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

Rajpurkar, P., Irvin, J., Bagul, A., Ding, D., Duan, T., Mehta, H., Yang, B., Zhu, K., Laird, D., Ball, R. L., et al. (2017). MURA: Large dataset for abnormality detection in musculoskeletal radiographs. *arXiv preprint arXiv:1712.06957*.

Rangayyan, R. M. (2004). *Biomedical Image Analysis*. CRC press.

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 779–788.

Reza, A. M. (2004). Realization of the contrast limited adaptive histogram equalization (clahe) for real-time image enhancement. *Journal of VLSI Signal Processing Systems for Signal, Image and Video Technology*, 38(1):35–44.

Ripley, B. D. (2007). *Pattern Recognition and Neural Networks*. Cambridge University Press.

Ronneberger, O., Fischer, P., and Brox, T. (2015). U-Net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 234–241. Springer.

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323(6088):533–536.

Sandfort, V., Yan, K., Pickhardt, P. J., and Summers, R. M. (2019). Data augmentation using generative adversarial networks (cycleGAN) to improve generalizability in CT segmentation tasks. *Scientific Reports*, 9(1):1–9.

Seeböck, P., Romo-Bucheli, D., Waldstein, S., Bogunovic, H., Orlando, J. I., Gerendas, B. S., Langs, G., and Schmidt-Erfurth, U. (2019). Using cycleGANs for effectively reducing image variability across OCT devices and improving retinal fluid segmentation. In *2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019)*, pages 605–609. IEEE.

Seide, F. and Agarwal, A. (2016). Cntk: Microsoft's open-source deep-learning toolkit. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2135–2135.

Srivastava, A., Valkov, L., Russell, C., Gutmann, M. U., and Sutton, C. (2017). VEEGAN: Reducing mode collapse in GANs using implicit variational learning. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan,

S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Stockman, G. and Shapiro, L. G. (2001). *Computer Vision*. Prentice Hall PTR, USA, 1st edition.

Stork, D. G., Duda, R. O., Hart, P. E., and Stork, D. (2001). Pattern classification. *A Wiley-Interscience Publication*.

Suetens, P. (2017). *Fundamentals of Medical Imaging*. Cambridge University Press.

Theano Development Team (2016). Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688.

Umbaugh, S. E. (2010). *Digital Image Processing and Analysis: Human and Computer Vision Applications with CVIPtools*. CRC press.

Wolfram Research (2019). AnatomyPlot3D, Wolfram language function. https://reference.wolfram.com/language/ref/AnatomyPlot3D.html. [Accessed: 16-March-2021].

Yadav, S. S. and Jadhav, S. M. (2019). Deep convolutional neural network based medical image classification for disease diagnosis. *Journal of Big Data*, 6(1):1–18.

Zhang, Z., Yang, L., and Zheng, Y. (2018). Translating and segmenting multimodal medical volumes with cycle-and shape-consistency generative adversarial network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9242–9251.

Zhou, S. K., Greenspan, H., and Shen, D. (2017). *Deep Learning for Medical Image Analysis*. Academic Press.

Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2223–2232.

Zhu, X., Liu, Y., Li, J., Wan, T., and Qin, Z. (2018). Emotion classification with data augmentation using generative adversarial networks. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 349–360. Springer.